Semester 1 Interim Group documentation

Table of Contents:

1. Project Title: Stagely Show Scraper

2. Project Overview

The goal of our project is to create a web scraping application that will take information about shows from all the theatre websites in the UK and push them to the Stagely CMS. The application will need to dynamically find the structure of these websites and extract the specific information Stagely wants to be on their CMS to streamline the process of adding a show to their system, the information we are looking to extract at a minimum is show name, production name, location, dates shown and a link to the show listing.

3.Project Scope

Our web scraping application will be scraping information about theatre shows in the UK. The scope of this includes professional and amateur theatres, but we will not be scraping information from other venues such as stadiums and cinemas.
The application will only be pushing information to the Stagely CMS so any other services it could be considered to interact with are out of scope

4. Project Objectives

- Dynamically read the structure of theatre websites to be scraped from
- Extract the relevant information from theatre websites and save it in an appropriate format
- Sort extracted information to only contain shows that are linked to a theatre production
- Sort information to be linked to a production as well as a theatre
- Push sorted information to the Stagely CMS to autofill fields streamlining the process of adding a show

5. Stakeholders

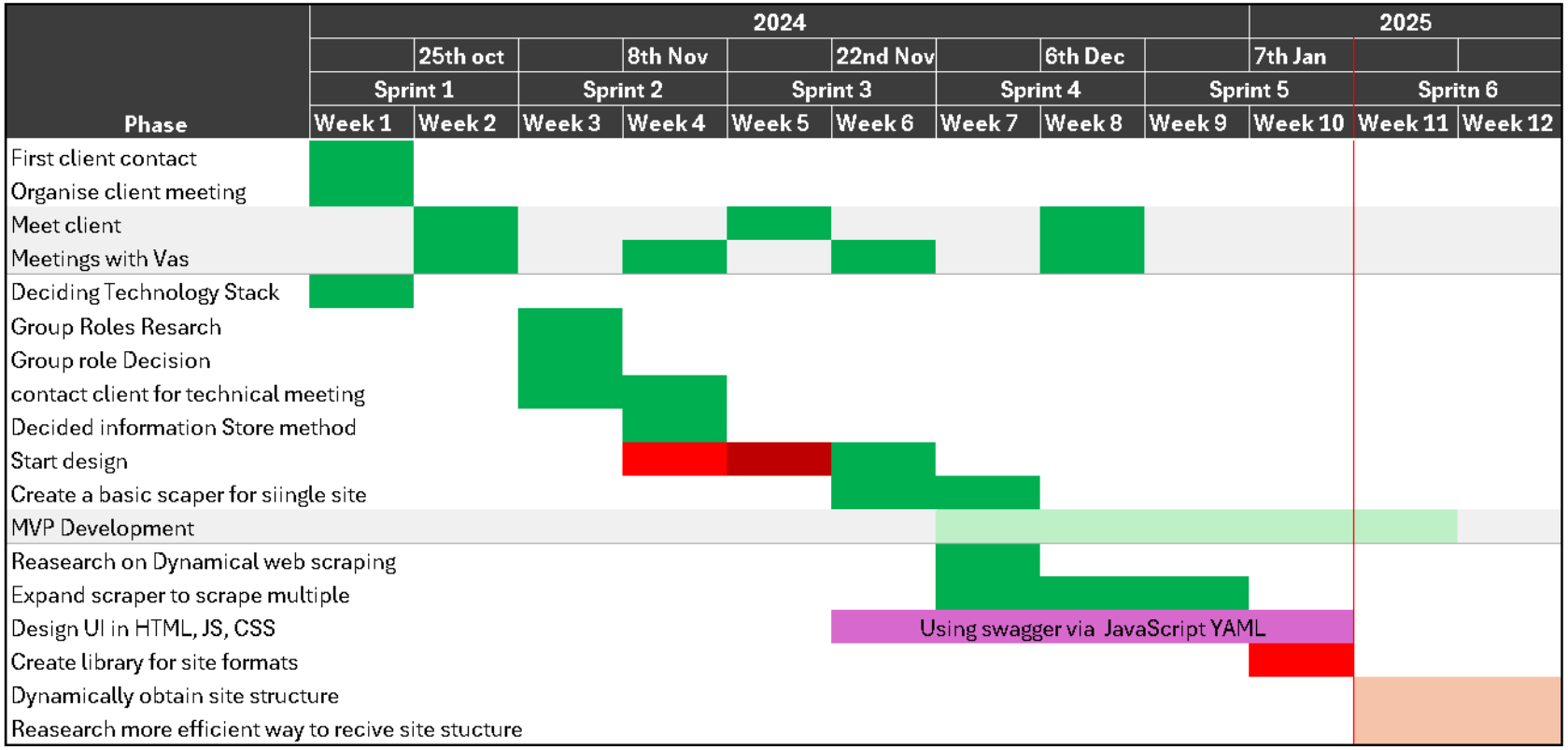
Stakeholder	Stagely LTD	Vasilis Cutsurdis	Stagely Users	Theatre Owners	Actors
Role	Client, Primary Internal Stakeholder	Scrum Master	Secondary External Stakeholder	External Stakeholder	External Stakeholders
Responsibilities	● Decide what he wants to be developed ● Create Ideas of features ● Review prototypes	● Review prototypes ● Help with issues between team and stakeholders ● Provide guidance	● View shows ● browse	● Not much responsibility unless there is copyright or GDPR involved ● Displaying correct information	● Make claim if they don't want their information shown/ scraped

6. Group: 10

Bence Bekefi	Bustamante Barrett-Greening	Jake Lear	Thomas Drew
- Architect	- Product Owner	- Team leader	- Scrum Master

7. Timeline - Gantt Chart

Below displays our gantt chart to show milestones and deadlines for key deliverables:



8. Research

We have done research into how to implement a JavaScript web scraper, finding that the most adaptable way to go about this is using Axios to make HTTP requests to the websites and Cheerio (Web scraping with Cheerio and Node.js) to parse the HTML of the websites to extract the relevant information we are looking for. Express.js can be used to build a web application capable of handling this functionality and to be able to run it locally we will use Node.js. This implementation can store the information in a way that can be interacted with as objects with attributes.

The section found at the bottom of the report covers LSEP responsibilities (Legal, Social, Ethical and professional), it also covers the continuity of the product, the sustainability and the risks.

9. Project Proposal

This document outlines the work we will be doing for Stagely LTD with a signature to signify their agreement.

We believe that we are the best choice to develop a web scraper to feed the "Stagely" Content Management System, demonstrated by a comprehensive SWOT analysis of our team and the project brief. Our strengths include previous experience using the programming languages REACT JS and C# within the Visual Studio environment, and an understanding of website design which will benefit in detecting the specific elements we want to scrape. We have worked together on coursework in our first year, so we already have the dynamics to excel as a team. The project aligns closely with our personal interests which will be motivating to provide the best product we can. Our weaknesses include a lack of prior experience in developing complex APIs and designing large databases so there will be a learning curve. However, through our current university modules and personal research we can develop these skills and apply them during the development of the web scraper.

This project also aligns with a gap in market demand which will improve access to regional theatre to current audiences and promote it to new audiences through a modern and accessible medium. There is the potential of partnering with regional theatres to integrate their databases into the web scraper to streamline data sharing and improve it's throughput by requiring less API calls.

Threats to this project are time management working on this alongside our other university modules coursework and the API maintenance as it will need to be updated whenever a theatre platform changes layout. User adoption must also be considered as theatre typically has an older target audience who are more cautious of new systems and reluctant to change from trusted services.

We believe through leveraging our strengths and taking advantages of the opportunities offered to us we will mitigate the weaknesses and threats posed to this project to ensure seamless execution of the web scraper development which will meet the required project features and provide useful and accurate information to the content management system.

After talking with the client, we are more confident in our ability to complete this project as we now know they have websites already categorized for scraping, saving research time and meaning we can get an initial version of the scraper working with their content manager sooner.

Daniel Lennox, Stagely Ltd

This is our signed project proposal, stating the work we will be doing for Stagely LTD and the co-founder (Daniel Lennox) has signed the bottom to acknowledge his agreement for us to take on this project for the company.

10. Work breakdown Structure

Chart of Epics

Epic	Description	WBS	Task
Epic 1: Requirements Gathering	Understand client needs, project objectives, and identify target theatre platforms for scraping.	1.1	Meet with the client to understand project objectives and features.
		1.2	Document categorized theatre websites for scraping.
		1.3	Identify key data fields required for the CMS (e.g., show names, dates, venues).
		1.4	Evaluate potential challenges with website structures (e.g., dynamic content, inconsistent HTML).
		1.5	Create a detailed project timeline and milestones.
		1.6	Make a draft project requirements documentation for client approval.
Epic 2: Web Scraper Development	Build the web scraper to extract data from targeted theatre websites and integrate with the CMS.	2.1	Research web scraping tools and libraries (Puppeteer).
		2.2	Develop scraper to parse HTML/CSS for required data fields.
		2.3	Configure scraper to handle dynamic content using JavaScript.
		2.4	Implement error handling for common issues (e.g., failed requests, misshaped HTML).
		2.5	Add logging and monitoring features for debugging and performance tracking.
		2.6	Optimize scraping process to minimize API calls and reduce data retrieval time.
Epic 3: API Design and Integration	Design APIs for data exchange and ensure seamless integration with the Content Management System.	3.1	Design RESTful API endpoints for data exchange between the scraper and CMS.
		3.2	Implement CRUD operations for managing scraped data.
		3.3	Test API endpoints for functionality, response time, and error handling.
		3.4	Secure APIs with authentication, rate limiting, and encryption protocols.
		3.5	Integrate API endpoints into the CMS for seamless data updates.
		3.6	Document API usage for future maintenance and scalability.
Epic 4: Database Design	Develop a database schema to store scraped data efficiently.	4.1	Design a relational database schema to store scraped data efficiently.
		4.2	Implement indexing for faster queries and data retrieval.
		4.3	Develop ETL (Extract, Transform, Load) pipelines for preprocessing scraped data.
		4.4	Test database performance under varying loads to ensure scalability.
		4.5	Implement validation rules and constraints to maintain data integrity.
		4.6	Create backup and recovery plans for the database.
Epic 5: Testing and Optimization	Perform testing of scraper functionality, optimize data extraction, and ensure accuracy of outputs.	5.1	Perform unit testing on individual scraper modules.
		5.2	Conduct integration tests for scraper, API, and database workflows.
		5.3	Optimize scraper performance to adapt to changes in website layouts.
		5.4	Validate data accuracy by comparing scraped data against original website content.
		5.5	Perform load testing on API and database to ensure stability under high usage.
		5.6	Implement feedback loops to address any issues found during testing.
Epic 6: User Training and Adoption	Provide training materials, workshops, and support to facilitate client and end-user adoption.	6.1	Develop user training materials, including guides and video tutorials.
		6.2	Conduct workshops or training sessions for client staff.
		6.3	Address digital divide, e.g. older audiences.
		6.4	Provide support for users during initial deployment.
		6.5	Gather feedback from users to improve usability and features.
Epic 7: Deployment and Maintenance	Deploy the scraper and API, monitor functionality, and plan for periodic updates and maintenance.	7.1	Deploy the scraper, API, and database to the production environment.
		7.2	Monitor performance of the scraper and address bugs or issues.
		7.3	Plan for regular updates to handle changes in theatre platforms' layouts.
		7.4	Set up a maintenance schedule for the scraper, database, and APIs.
		7.5	Develop a version control strategy for future enhancements.
		7.6	Create a post-deployment review report for the client.

11. Resource Plan

The below table discusses our resource plan, of human, technical and cost. The plan runs of of the Epics produced above

Epic	Resource Type	Details	Cost (GBP)
Epic 1: Requirements Gathering	Human Resources	Project Leader, Product Owner, Client	£0
	Technical Resources	Zoom, Google Docs, or Microsoft Teams	Free
	Financial Resources	Minimal subscription fees	£0–£20
Epic 2: Web Scraper Development	Human Resources	Developer, Project Leader	£0
	Technical Resources	Puppeteer	Free
		VS Code	Free
	Financial Resources	No additional costs	£0
Epic 3: API Design and Integration	Human Resources	Developer, Tester, Project Leader	£0
	Technical Resources	Flask,Node.JS	Free
		Postman for API testing	Free
		Hosting for API (AWS/Heroku/ University)	£0–£50/month
	Financial Resources	Potential hosting upgrade for production	£50/month
Epic 4: Database Design	Human Resources	Developer, Project Leader, architecht	£0
	Technical Resources	SQLServer, Azure datastudio, or SQLite	Free
	Financial Resources	No additional costs	£0
Epic 5: Testing and Optimization	Human Resources	Tester, Developers	£0
	Technical Resources	Pytest, JMeter, or similar tools	Free
		Test environments (local servers)	Free
	Financial Resources	No additional costs	£0
Epic 6: User Training and Adoption	Human Resources	Designer, Project Leader, Architecht	£0
	Technical Resources	Canva, Google Docs, YouTube (free versions)	Free
	Financial Resources	Premium design tools for training materials	£0–£50
Epic 7: Deployment and Maintenance	Human Resources	Developers	£0
	Technical Resources	Hosting: AWS, Azure, or Heroku (cant use Uni after deployment	£50–£100/month
	Financial Resources	Hosting	£50–£100/month

12. Risk management

Risk management is featured in the Risk section of the LSEP

13. Communication plan

Our communication plan for passing on the project at the end of the course is detailed in the Continuity section of LSEP
Across the team we will strive for clear, consistent, and efficient communication among team members and with the client throughout the project. We are and will continue to communicate in a combination of meeting calls, emails, reports and in person meetings, as these methods allow flexibility. Every other week we meet with the client and in the weeks that we don't meet the client, we have our scrum meeting where we can discuss progress and map the path for the near and far future.
As a team we have been using discord to communicate between each other, allowing us to do video calls, share files, share code blocks and also develop together.

14. Quality management

To ensure the quality of our work is kept to the highest standard we will be thoroughly testing the application at every stage of development to ensure the outputs are as expected and there are no bugs which could cause issues further down the development timeline.

The code for our application will follow good coding practice standards and be well commented to make it readable which will ensure the end product is of high quality and up to the clients standards.

15. Monitoring and Evaluation

To monitor and evaluate the progress of our project we are working in 2-week sprints with meetings with Vasilis Curtsurdis and Daniel Lennox seperately on rotating weeks to report the progress we have made and receive any necessary feedback on how our progress is coming along. We have been taking minutes of our meetings for us to use to evaluate what we've done well and what we need to work on for the next sprint.

16. Budget

There is currently no budget for the project and we are aiming to use free services if we need to use any to make our application fully functional. Although the client has stated they are open to suggestions if there is a paid service, for example those mentioned in the Resource Plan like AWS for hosting the API and Scraper

17. Approval

Client has provided a signature to state they have approved for us to take on this project and they are updated in our sprint meetings of progress where they can provide feedback to ensure the project aligns with their expectations

18. Change management

We have already managed changes in our project spec with the changing primary focus from being a mobile application for Stagely LTD to a web scraping application that will interface with the Stagely LTD content management system. To manage this change we adapted our client proposal to meet the new requirements and expectations of Stagely then adjusting our roles to better fit the new requirements.
Should there be any more changes to the project specification we will adapt our documentation to reflect this change and make any necessary edits to the application, we believe the direction of the project to be more concrete now so any changes that we are given should not cause a great impact on our progression.

19. Closure and evaluation

Moving forward we will be expanding the application to be able to dynamically scan the structure and extract information from every site Stagely has requested rather than the small set it is currently functional for.
We now need to be implementing functionality to sort the data we retrieve and push it to the Stagely CMS in a format that they expect so that the application will be useful to Stagely for the purpose they have envisioned it for.

In semester 2 we will need to manage our time and keep our documents updated much better alongside other modules as we have failed to do this in semester 1, meaning that we have had to backdate important project management sections such as our Trello boards. We will also need to ensure all group members are sharing the workload and supported to fulfill their personal roles.

We will also commit to the github more regularly instead of uploading all of the documents at once as this will keep a more professional approach and help us to adhere to agile better.

Separate to Plan: LSEP

Legal

Our Project involves a web scraper, which will be scraping data off of public websites, as different websites have different layouts and setups we have to be careful what it is we scrape. We will not be dealing with user data or personal data that is not public, so as long as we develop the application carefully there will be no chance of GDPR breaches. We will be scraping from theater websites which will fall under the category of “Products data from eCommerce sites for competitor tracking and price intelligence,” (Barrett) which according to Octoparse, is legal to scrape. However the Scraper will also be getting details of actors in the shows, according to Octoparse “if you’re scraping name, age, location, and other details too then you’re entering into the PII zone and need to be GDPR compliant for addressing legal compliance.” (Barrett) and so with personal data like this we can obtain consent, ensure the data is truly public and avoid sensitive data.

Another legal factor to be aware of is copyright, when scraping details, for example of a performance, we would need to ensure that there is permission to take such information, like

images or descriptions of the performance. References will be the key to copyright, to ensure that we are compliant.

Ethical

The irrespective collection or data would be unethical as “... a computing professional should become conversant in the various definitions and forms of privacy and should understand the rights and responsibilities associated with the collection and use of personal information." (“ACM Code of Ethics and Professional Conduct”), as mentioned in Legal, it is our responsibility to be attentive and concerned about the data that we collect.

We will need to maintain clarity with all stakeholders, ensuring that the sources of data collection are known where necessary. The data collected will need to be fairly represented and not provide the wrong image of a show or venue, incorrectly displaying data would be unethical as we are misinforming and misrepresenting.

Social

Due to the digital divide (“COVID-19 and the digital divide - POST”) there may be users that are using the CMS interface that are non-technical and henceforth the CWM interface will need to be easy to use and navigate, this is displayed later in the document with our Figma. Ensuring an intuitive interface will bridge the accessibility gaps so many users can use our system. For users with disabilities we could also incorporate a screen reader to ease their use of the app.

Our scraper will be scraping all productions from all theatres in the Uk and so this will allow for smaller theatres to compete on a fair-ground against the industry greats like Shakespeare's globe or the royal albert hall (Isle) as well as other reputable west end theatres. Giving more publicity to the smaller theatres will help people interested in theatre will less money, be able to see shows, therefore widening the accessibility to theatre, but not only this, it will allow small theatres to build their customer base and reputation leveling the playing fields.

Another feature is the scrapping tags of the theatres, having tags like “accessible” or “family-friendly” will allow the user base of Stagely to choose their shows based on certain needs.

Professionalism

To provide a professional service, the scraper will need to be run often so that users of Stagely will always have accurate information about current shows and availability as well as local theatres.
There is also a professional responsibility to ensure data integrity. Computation integrity means “data is complete, trustworthy and has not been modified or accidentally altered by an unauthorised user. The integrity of data can be compromised unintentionally by errors in entering data, a system malfunction, or forgetting to maintain an up-to-date backup.” (“What Is Integrity In Cyber Security?”) To conform to this there will need to be security measures in

the CMS to prevent unauthorised users defiling the integrity of the data. We will also need to ensure that the sites we are scraping from are legitimate.

The quality of our development also needs to be of a professional quality as it is being developed for a client's use in a real world service.

With the web scraper there will be documentation of instructions concerning the deployment of the scraper, to mitigate the risks of GDPR breaches or compromise to the integrity of data.

Risks

The scope of the project is quite wide, with us having to make scrapers for the whole of the Uk's theatre industry, and so there are risks involved in the timeline. To effectively develop the project we are keeping to agile development philosophies so we can complete the project in the time frame.

Another risk is GDPR, however these risks are discussed in its section.

Another risk is websites updating their source code and HTML, this could have potential for nullifying how our scraper module will work on the site, and due to this we will need to keep constant monitoring as well as health checks in the scraper that flag issues and where.

Sustainability

The code needs to be developed in a modular way so it can be repurposed for different scraping tasks. To ensure this, the final development will encompass the DRY development principle. ""Don't Repeat Yourself." Living by this principle means that your aim is to reduce repetitive patterns and duplicate code and logic in favor of modular and referenceable code" (Poppy).

We will also minimise resource use by limiting redundant operations and also using optimisation.

Continuity

Once the project is finished, the client will still need to be able to use the scraper so we will need to provide details on:

- Instructions on deploying the scraper
- Updating configurations
- Troubleshooting common errors.

Using GitHub will allow future developers to access older versions to see how it has been developed and we will also make sure that the final product is easy to extend through the use of modularity and implementations of comments + DRY and SOLID principles. (Levy)

Minimum Viable Product

We now have an initial version of the web scraper working on a few theatres. To make it be able to dynamically read the contents of multiple sites, we have created getStructure.js which reads a JSON list of websites (currently 25) and scans them for the expected selectors which are fetched from potentialSelectors.json and then saves the found website structures to websiteConfigs.json.
However, It is currently only able to recognise the structure of 8 of these websites due to the limitation of needing the specific site selectors in potentialSelectors.json, we are now at the stage where we need to decide if our current approach is scalable (have specific selectors saved for every site we want to scan) or whether there is a more suitable way to dynamically retrieve the site specific selectors to be used to scrape the productions.
The scraper.js file holds the logic for the scrape endpoint and reads the websiteConfigs.json file to work through all the successfully scanned sites by reading the contents of the provided elements and return the current shows with details being held at the theatres by saving them in fetched_shows.json.
Moving forward we will be deciding the best approach to dynamically retrieve site layouts and will be expanding our application to take the fetched_shows output and push it to the Stagely LTD content management system. Ideally we will also be able to retrieve more information from the sites as currently most of them only store the show title, dates it's on and the show link.

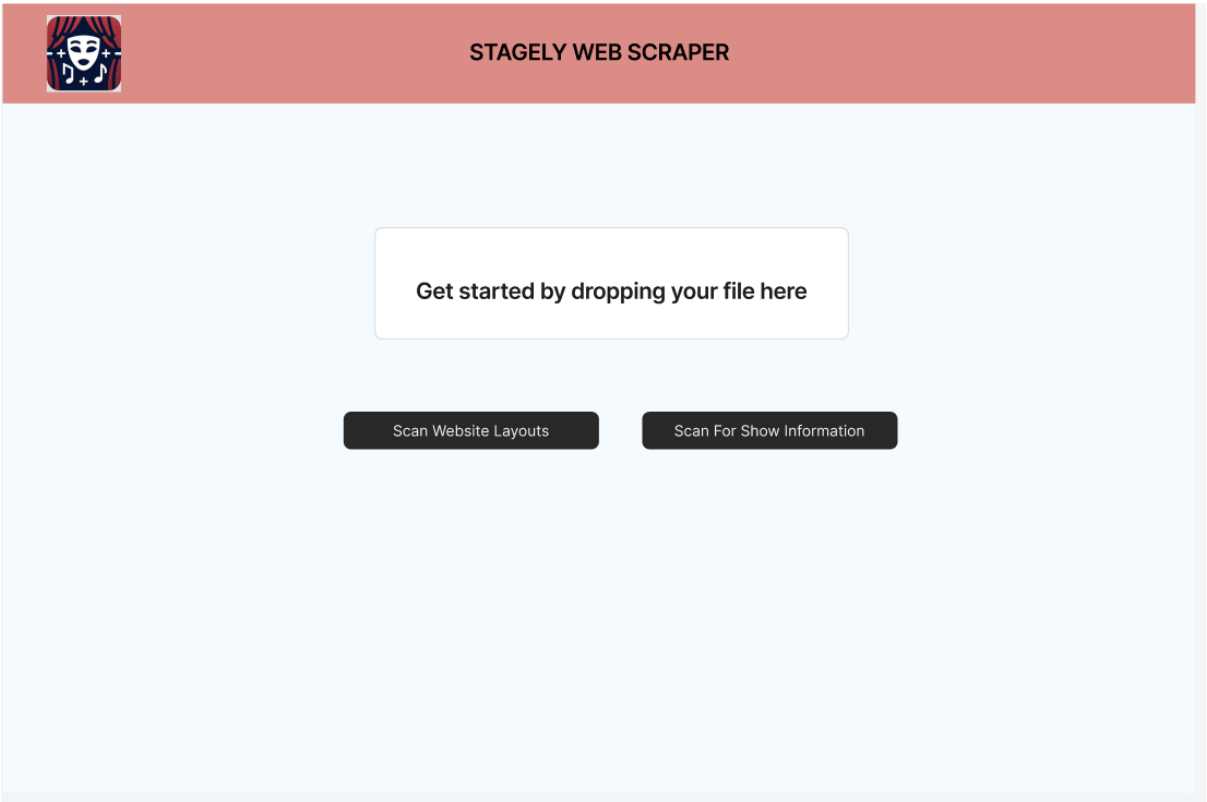
Source Code is in the document appendix

GitHub Link:

https://github.com/Plymouth-University/comp2003-2024-group-10

20. Appendix

1. Application interface design



2. Personal Reflection Links

Name	Link
Bence Bekefi	https://drive.google.com/file/d/1jfH3tKZYaaOUykkL4W-jtO1lK0vCZmyJ/view?usp=sharing
Bustamante Barret-Greening	https://drive.google.com/file/d/1_bQmpKejXA9Zl9O_6vncyTGSv02-n-5H/view?usp=sharing
Jake Lear	https://drive.google.com/file/d/1sPDdbcKk1LhxK27jLrJIFuXImP9zpnLh/view?usp=sharing
Thomas Drew	https://drive.google.com/file/d/1VY2Q8QpC1rr_TcrXJMkNVim0_VaxeIGZ/view?usp=sharing

3. Works Cited

"ACM Code of Ethics and Professional Conduct." *Association for Computing Machinery*, https://www.acm.org/code-of-ethics. Accessed 6 January 2025.

Barrett, Ansel. "GDPR Compliance In Web Scraping." *Octoparse*, 15 August 2021, https://www.octoparse.com/blog/gdpr-compliance-in-web-scraping. Accessed 6 January 2025.

"COVID-19 and the digital divide - POST." *POST Parliament*, Susie Wright, 17 December 2020, https://post.parliament.uk/covid-19-and-the-digital-divide/. Accessed 6 January 2025.

Isle, Paul -. "THE 10 BEST London Theatres (2025)." *Tripadvisor*, https://www.tripadvisor.co.uk/Attractions-g186338-Activities-c58-t116-London_England.html. Accessed 6 January 2025.

Levy, Jean. "Principles of Software Development: SOLID, DRY, KISS, and more." *Scalastic* , 1 July 2023, https://scalastic.io/en/solid-dry-kiss/. Accessed 6 January 2025.

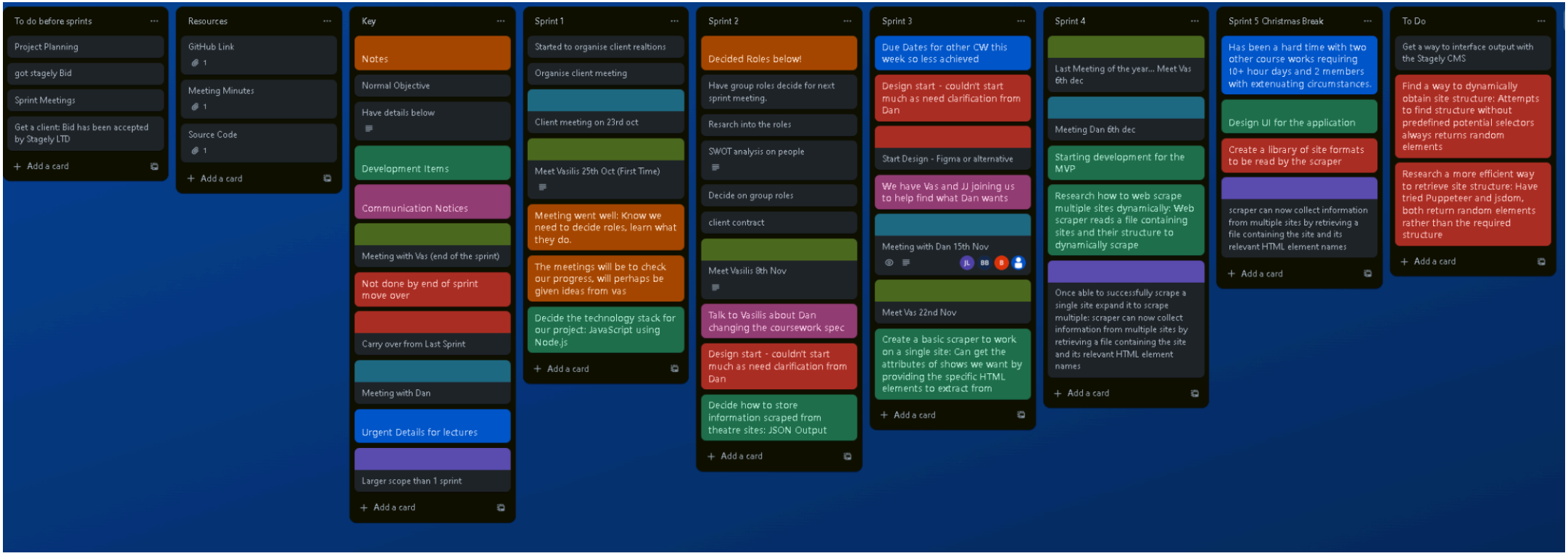
Poppy, Daniel. "Guide to DRY." *dbt Labs*, 1 July 2024, https://www.getdbt.com/blog/guide-to-dry. Accessed 6 January 2025.

Web scraping with Cheerio and Node.js, Waweru. "Web scraping with Cheerio and Node.js." *Web scraping with Cheerio and Node.js*, 14 09 2023, https://circleci.com/blog/web-scraping-with-cheerio/. Accessed 07 01 2025.

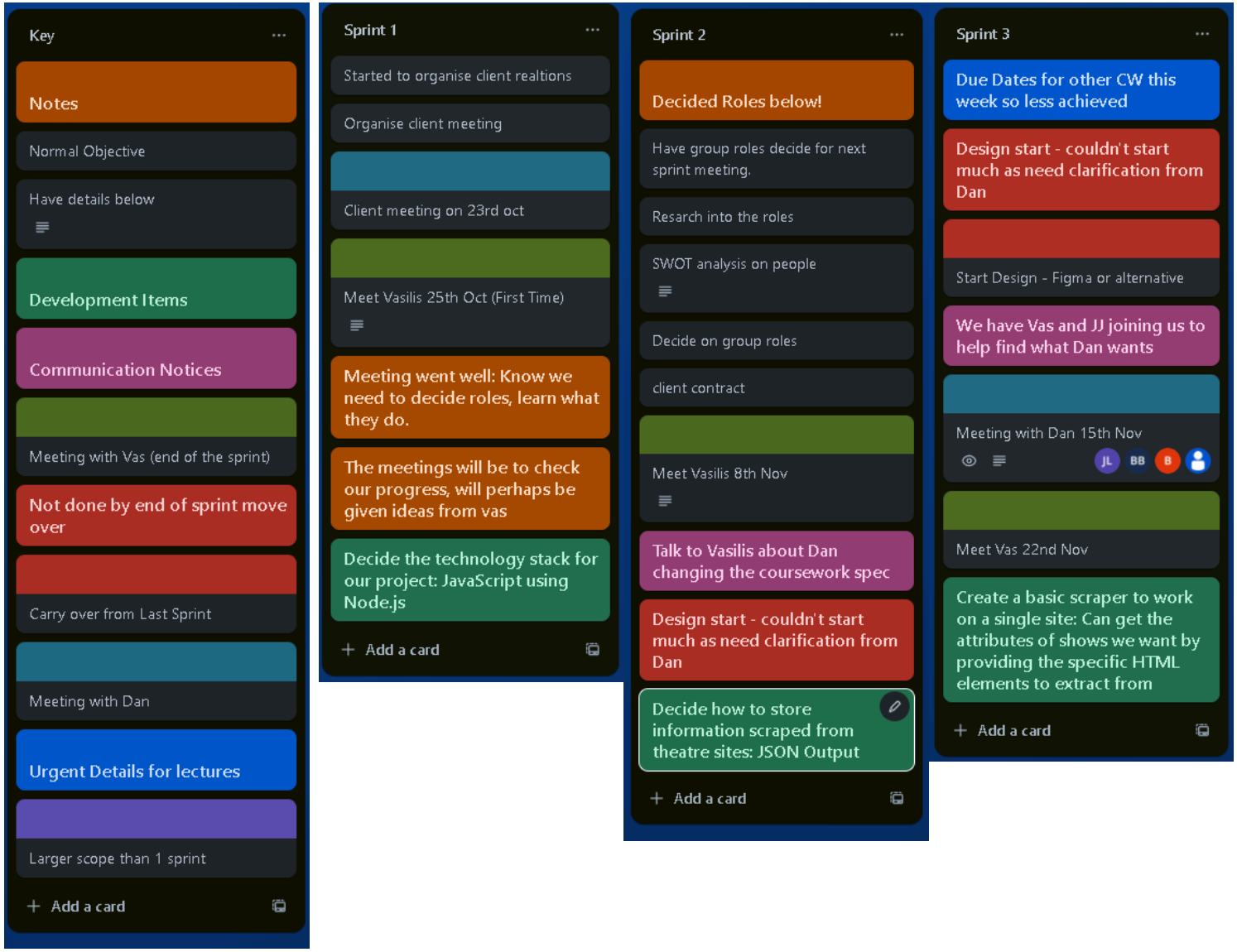
"What Is Integrity In Cyber Security?" *RiskXchange*, https://riskxchange.co/1006895/what-is-integrity-in-cyber-security/. Accessed 6 January 2025.

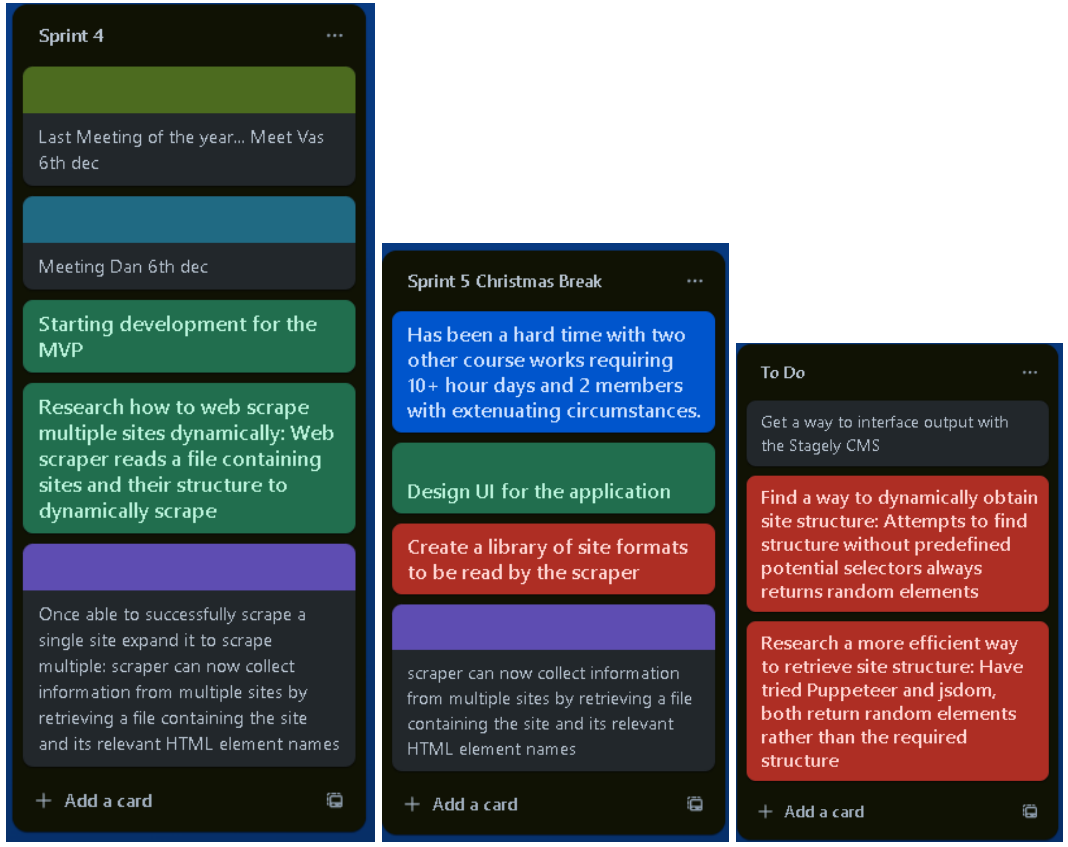
4. Kanbans

Main sprint kanban as whole:

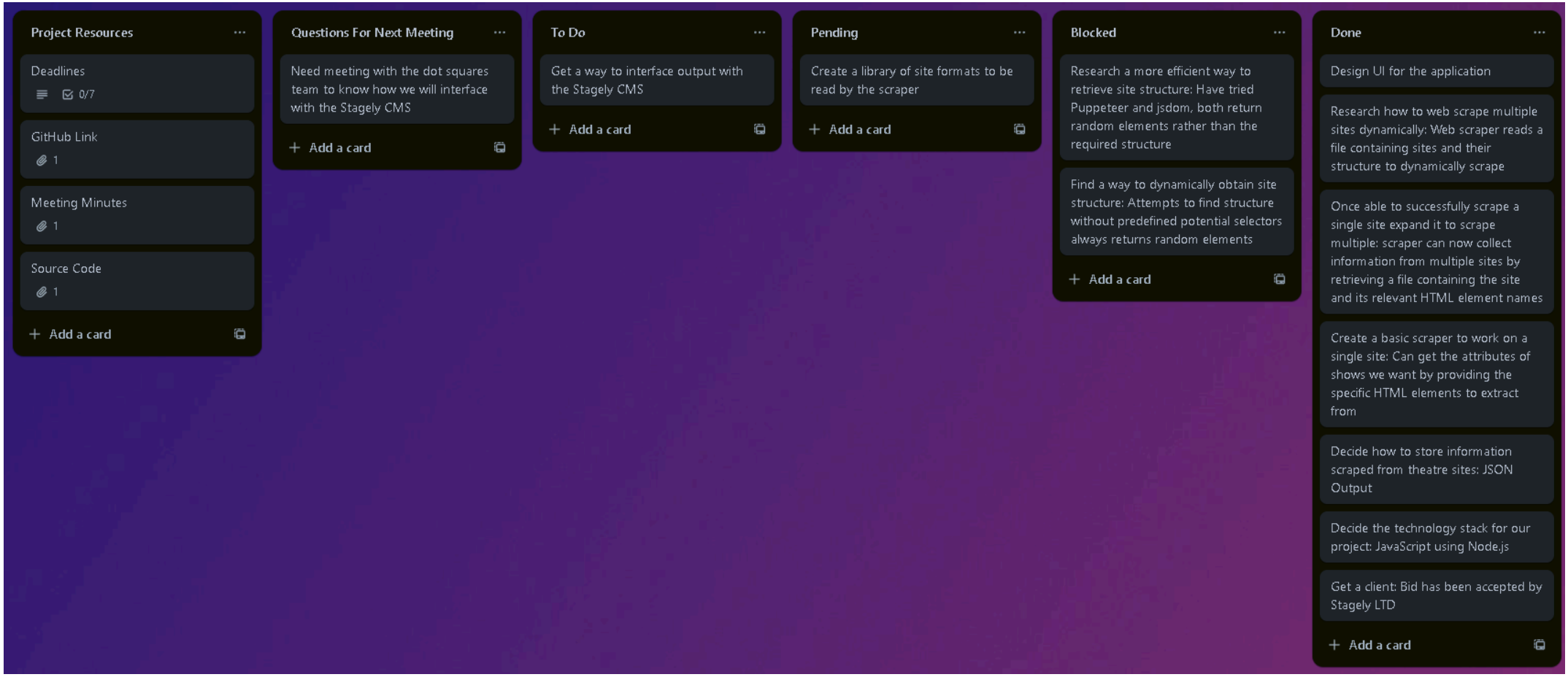


Better Views of kanbans

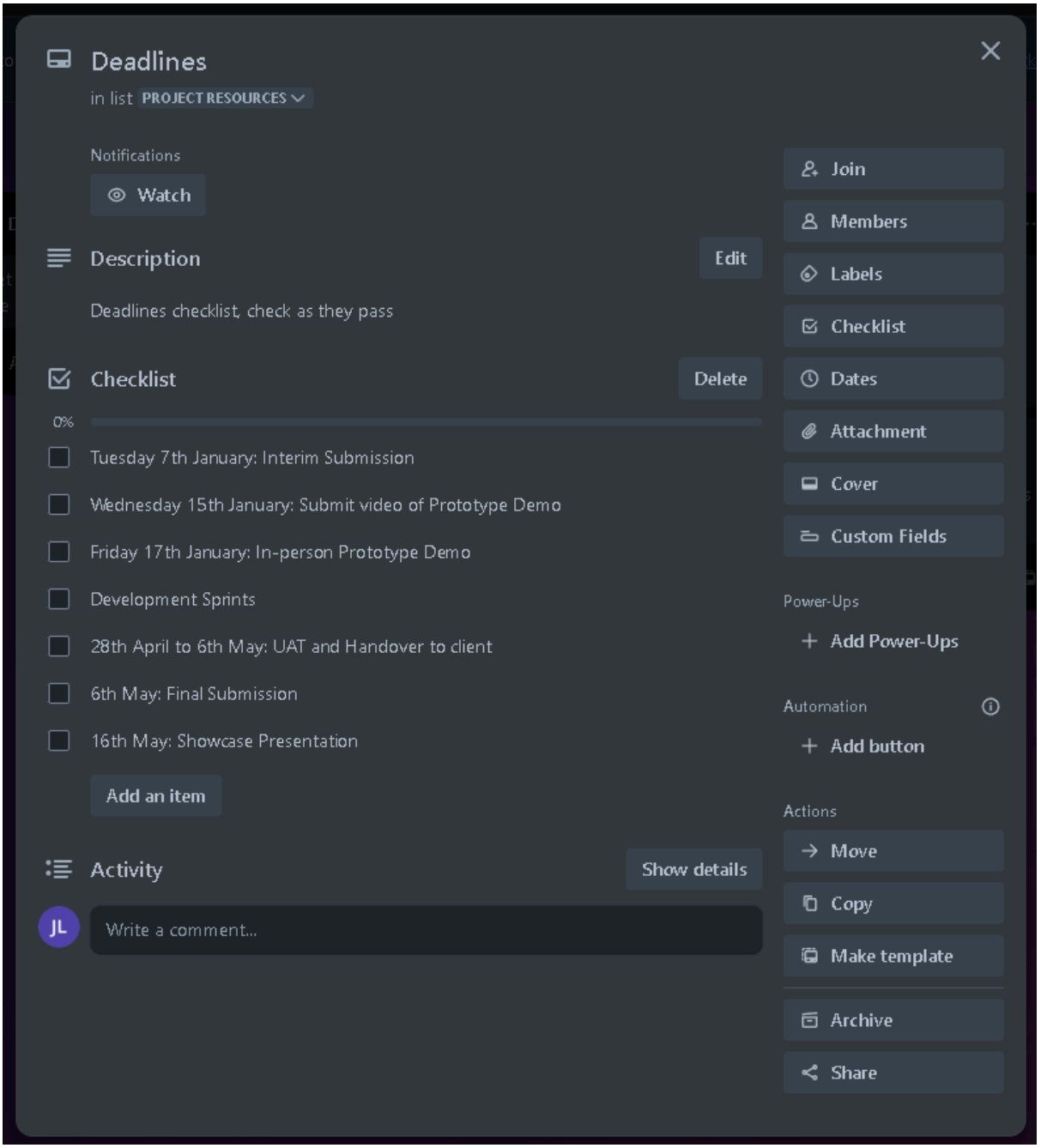



Sprint 4
Last Meeting of the year... Meet Vas 6th dec
Meeting Dan 6th dec
Starting development for the MVP
Research how to web scrape multiple sites dynamically: Web scraper reads a file containing sites and their structure to dynamically scrape
Once able to successfully scrape a single site expand it to scrape multiple: scraper can now collect information from multiple sites by retrieving a file containing the site and its relevant HTML element names
Add a card
Sprint 5 Christmas Break
Has been a hard time with two other course works requiring 10+ hour days and 2 members with extenuating circumstances.
Design UI for the application
Create a library of site formats to be read by the scraper
scraper can now collect information from multiple sites by retrieving a file containing the site and its relevant HTML element names
Add a card
To Do
Get a way to interface output with the Stagely CMS
Find a way to dynamically obtain site structure: Attempts to find structure without predefined potential selectors always returns random elements
Research a more efficient way to retrieve site structure: Have tried Puppeteer and jsdom, both return random elements rather than the required structure
Add a card

Project management Kanban:



Project management kanban deadlines tab:



5. Project plan checklist

Group 10 COMP2003.xlsx

Project Plan Structure	Done?
1. Project Title: Stagely Project	Y
2. Project Overview: Create a web scraper to take information from theatre production sites and place it into a content management system	Y
Project Scope: Define the boundaries of the project, specifying what is included and excluded.	Y
4. Project Objectives: Clearly state the measurable and achievable outcomes the project aimsto accomplish.	Y
5. Stakeholders: Identify and list all stakeholders involved in the project, including their roles and responsibilities.	Y
6. Project Team: Outline the members of the project team, their roles, and reporting relationships.	Y
7. Timeline: Create a detailed timeline with milestones and deadlines for key deliverables. Use a Gantt chart to illustrate this clearly.	Y
8. Research: present findings from existing work that address the problem statement, scope and objectives of the project. Main part of this section is to address existing competition and solutions, and how your project is unique in its approach. You may pull material from your design document for this.	Y
9. Proposed Solution: an overview of your project's solution and methodologies. This should be in line with your overview, scope and, objectives, timeline and further elaborated on under the Work Breakdown Structure next.	Y

10. Work Breakdown Structure (WBS): Break down the proposed solution into smaller, manageable tasks and subtasks. Create a hierarchical structure showing the relationship between different tasks. You may pull material from your design document for this.	Y
11. Resource Plan: Identify the resources (human, financial, equipment) required for each task and allocate them accordingly.	Y
12. Risk Management: Identify potential risks and develop strategies for risk mitigation. Include contingency plans for addressing unforeseen issues.	Y
13. Communication Plan: Define how communication will be handled throughout the project, including regular meetings, reporting mechanisms, and channels of communication.	Y
14. Quality Management: Specify the quality standards and processes that will be used to ensure the project's deliverables meet the required criteria.	Y
15. Monitoring and Evaluation: Outline how the project's progress will be monitored and evaluated, including key performance indicators (KPIs).	Y
16. Budget: Provide a detailed budget outlining the costs associated with the project, including resources, materials, and any other relevant expenses. Cost savings using open-source solutions that are free are a big plus point to be mentioned here.	Y
17. Approval Process: Clearly define the process for obtaining approvals at different stages of the project. For this module, you will require the client’s approval/signoff for this.	Y
18. Change Management: Describe how changes to the project scope, schedule, or resources will be identified, evaluated, and implemented. This is in case the client wants to request changes in semester 2, what’s your process for doing so?	Y

19. Closure and Evaluation: Outline the steps for closing out the project, including a postimplementation review and lessons learned. This should be in line with the final submission checklist provided in the Handbook.	Y
20. Appendices: Include any additional documentation or reference materials that support the project plan	Y

Sourcecode

server.js

```
const express = require("express");
const bodyParser = require("body-parser");
const swaggerUi = require("swagger-ui-express");
const yaml = require("js-yaml");
const fs = require("fs");
const path = require("path");

const scrapeProducts = require("./scraper");
const { processWebsitesFromFile } = require("./getStructure");

const app = express();
const port = process.env.PORT || 3000;

// Middleware setup
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// Load Swagger YAML file for UI
const swaggerDocument = yaml.load(fs.readFileSync(path.join(__dirname,
"swagger.yaml"), "utf8"));

// Serve Swagger UI
app.use("/api", swaggerUi.serve, swaggerUi.setup(swaggerDocument));

// Route for scraper.js endpoint
app.use("/", scrapeProducts);

// Route for getStructure.js endpoint
app.post("/fetchStructure", async (req, res) => {
  try {
    const results = await processWebsitesFromFile();
    res.status(200).json({
      message: "Website structures processed successfully.",
      results,
    });
  } catch (error) {
    console.error("Error processing websites:", error.message);
    res.status(500).json({ error: error.message });
  }
});
```

```
// Start the server and state port and swagger link
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
  console.log(`Swagger docs available at
http://localhost:${port}/api`);
});
```

scraper.js

```
const express = require("express");
const cheerio = require("cheerio");
const axios = require("axios");
const fs = require("fs");
const path = require("path");
const router = express.Router();
const { generateFilename } = require("./utils");

const loadWebsiteConfigs = () => {
  const websiteConfigsPath = path.join(__dirname,
"../docs/websiteConfigs.json");
  if (!fs.existsSync(websiteConfigsPath)) {
    throw new Error(`File ${websiteConfigsPath} does not exist.`);
  }
  return JSON.parse(fs.readFileSync(websiteConfigsPath, "utf-8"));
};

router.get("/scrape", async (req, res) => {
  const results = [];
  const scrapedData = [];
  let websiteConfigs;

  try {
    websiteConfigs = loadWebsiteConfigs();
  } catch (error) {
    return res.status(500).json({ error: `Failed to load website
configurations: ${error.message}` });
  }

  for (const site of websiteConfigs) {
    const { name, url, selectors } = site;

    try {
      const response = await axios.get(url);
      const $ = cheerio.load(response.data);
```

```
      const events = [];

      // Extract theatre name dynamically for ATG sites from the URL
      let theatreName = name; // Default to site name
      if (url.includes("atgtickets.com")) {
        const match = url.match(/\/venues\/([^\/\?]+)/); // Extract the
part after "/venues/" for ATG theatres
        theatreName = match ?
decodeURIComponent(match[1]).replace(/-/g, " ").replace(/\b\w/g, (c) =>
c.toUpperCase()) : name; // Format as title case
      }

      $(selectors.eventCard).each((i, el) => {
        const event = $(el);

        const title = event.find(selectors.title).text().trim() || "";
        const date = event.find(selectors.date).text().trim() || "";
        const location = selectors.location ?
event.find(selectors.location).text().trim() : "";
        const linkElement = event.find(selectors.link);
        const link = linkElement.length > 0 ? linkElement.attr("href")
: "";

        const fullLink = link && link.startsWith("http") ? link : new
URL(link, url).href;

        if (title && date && fullLink) {
          events.push({
            title,
            date,
            location,
            link: fullLink,
          });
        }
      });

      scrapedData.push({
        theatre: theatreName,
        shows: events,
      });

      results.push({
```

```
        site: name,
        events_saved: events.length,
        message: "Scraped successfully",
      });
    } catch (error) {
      console.error(`Error scraping ${name}:`, error.message);
      results.push({
        site: name,
        events_saved: 0,
        message: `Error scraping: ${error.message}`,
      });
    }
  }

  // Save all the scraped data into a single file with the date, time
and scraped shows suffix
  const outputFile = path.join(
    __dirname,
    "../dataOutput",
    generateFilename(".json", "-scraped-shows")
  );
  fs.writeFileSync(outputFile, JSON.stringify(scrapedData, null, 2));

  res.json({
    message: "Scraping completed successfully.",
    results,
    outputFile,
  });
});

module.exports = router;
```

getStructure.js

```
const axios = require("axios");
const cheerio = require("cheerio");
const fs = require("fs");
const path = require("path");

/**
 * Load potential selectors from the JSON file.
 * @returns {object} - Potential selectors for event cards.
 */
const loadPotentialSelectors = () => {
```

```
  const selectorsFile = path.join(__dirname,
"../docs/potentialSelectors.json");
  if (!fs.existsSync(selectorsFile)) {
    throw new Error(`File ${selectorsFile} does not exist.`);
  }
  return JSON.parse(fs.readFileSync(selectorsFile, "utf-8"));
};

/**
 * Validate selectors by checking for their existence on the page.
 * @param {object} $ - Cheerio instance of the page content.
 * @param {string[]} potentialSelectors - List of potential selectors
to check.
 * @returns {string|null} - The first matching selector, or null if
none found.
 */
const findMatchingSelector = ($, potentialSelectors) => {
  for (const selector of potentialSelectors) {
    if ($(selector).length > 0) {
      return selector;
    }
  }
  return null;
};

/**
 * Analyze the webpage for event structure using potential selectors.
 * @param {object} $ - Cheerio instance of the page content.
 * @param {object} potentialSelectors - Potential selectors for event
cards.
 * @returns {object|null} - Detected selectors for the event structure,
or null if none found.
 */
const detectEventStructure = ($, potentialSelectors) => {
  const eventCard = findMatchingSelector($,
potentialSelectors.eventCard);
  if (!eventCard) return null;

  const title = findMatchingSelector($, potentialSelectors.title);
  const date = findMatchingSelector($, potentialSelectors.date);
  const location = findMatchingSelector($,
potentialSelectors.location);
  const link = findMatchingSelector($, potentialSelectors.link);
```

```
  if (title && date && link) {
    return { eventCard, title, date, location, link };
  }

  return null;
};

/**
 * Analyze a single website to detect event structure.
 * @param {string} url - The URL of the website to analyze.
 * @param {object} potentialSelectors - Potential selectors for event
cards.
 * @returns {object|null} - The detected configuration or null if
analysis fails.
 */
const analyzeWebsite = async (url, potentialSelectors) => {
  try {
    console.log(`Analyzing ${url}...`);
    const response = await axios.get(url);
    const $ = cheerio.load(response.data);

    // Detect event structure dynamically
    const selectors = detectEventStructure($, potentialSelectors);
    if (!selectors) {
      console.warn(`No valid structure found for ${url}`);
      return null;
    }

    return {
      name: new URL(url).hostname,
      url,
      selectors,
    };
  } catch (error) {
    console.error(`Error analyzing ${url}: ${error.message}`);
    return null;
  }
};

/**
 * Processes websites listed in the `websites.json` file and saves
configurations to `websiteConfigs.json`.
```

```
 * @returns {object} - Object containing results and total counts.
 */
const processWebsitesFromFile = async () => {
  const websitesFile = path.join(__dirname, "../docs/websites.json");
  const outputFile = path.join(__dirname,
"../docs/websiteConfigs.json");

  // Load potential selectors
  const potentialSelectors = loadPotentialSelectors();

  // Check if the input file exists
  if (!fs.existsSync(websitesFile)) {
    throw new Error(`File ${websitesFile} does not exist.`);
  }

  // Read the list of websites
  const websites = JSON.parse(fs.readFileSync(websitesFile, "utf-8"));
  if (!Array.isArray(websites) || websites.length === 0) {
    throw new Error(`No websites found in ${websitesFile}.`);
  }

  const results = [];
  for (const url of websites) {
    const config = await analyzeWebsite(url, potentialSelectors);
    if (config) {
      results.push(config);
    }
  }

  if (results.length > 0) {
    fs.writeFileSync(outputFile, JSON.stringify(results, null, 2));
    console.log(`Website configurations saved to ${outputFile}`);
  }

  return {
    message: "Website structures processed successfully.",
    totalWebsites: websites.length,
    successfulScrapes: results.length,
    savedStructures: results,
  };
};
```

```
// Run the function and export its result for use in the scraping
endpoint
if (require.main === module) {
  processWebsitesFromFile()
    .then((result) => {
      console.log(result);
    })
    .catch((err) => console.error("Error:", err.message));
}

module.exports = { processWebsitesFromFile };
```

utils.js

```
const fs = require("fs");

// Generate unique filenames with an optional extension
const generateFilename = (extension = ".json") => {
  const date = new Date();
  const year = date.getFullYear();
  const month = date.getMonth() + 1;
  const day = date.getDate();
  const hour = date.getHours();
  const minute = date.getMinutes();
  const second = date.getSeconds();
  const filename =
`${year}-${month}-${day}-${hour}-${minute}-${second}${extension}`;
  return filename;
};

// Save the raw HTML to a file (for debugging to ensure the selectors
are correct)
const saveHtml = (html) => {
  const filename = generateFilename(".html");
  // If env == test create a test data folder, else create a
scrapedHTML folder
  const folder = process.env.NODE_ENV === "test" ? "test-data" :
"scrapedHTML";
  // Create a new folder if it doesn't exist
  if (!fs.existsSync(folder)) {
    fs.mkdirSync(folder);
  }
  fs.writeFileSync(`${folder}/${filename}`, html);
  return `./${folder}/${filename}`;
};
```

```
// Export functions for use in scraper
module.exports = {
  generateFilename,
  saveHtml,
};
```

swagger.yaml

```
openapi: 3.0.0
info:
  title: Show Scraper API
  description: API for scraping product and event data
  version: 1.0.0
servers:
  - url: http://localhost:3000
paths:
  /scrape:
    get:
      summary: Automatically scrapes shows, location, and dates from
predefined sites.
      responses:
        '200':
          description: A JSON object with the number of events saved
per site.
          content:
            application/json:
              schema:
                type: object
        '500':
          description: Error scraping events.
  /fetchStructure:
    post:
      summary: Processes websites to detect their structure and saves
the configuration.
      description: This endpoint reads a predefined list of websites
and writes the results to a configuration file.
      responses:
        '200':
          description: Successfully processed website structures.
          content:
            application/json:
              schema:
                type: object
                properties:
```

```
                message:
                  type: string
                results:
                  type: array
                  items:
                    type: object
                    properties:
                      name:
                        type: string
                      url:
                        type: string
                      selectors:
                        type: object
                        properties:
                          eventCard:
                            type: string
                          title:
                            type: string
                          date:
                            type: string
                          location:
                            type: string
                          link:
                            type: string
        '500':
          description: Error processing website structures.
```

JSON documents

Files read by the JavaScript code to get information from theatre websites

websites.json

```
[
"https://www.shakespearesglobe.com/",
"https://www.atgtickets.com/venues/lyceum-theatre/?utm_source=google&utm_medium=organic&utm_campaign=gmb",
"https://lwtheatres.co.uk/theatres/his-majestys/",
"https://www.atgtickets.com/venues/apollo-victoria-theatre/?utm_source=google&utm_medium=organic&utm_campaign=gmb",
"https://lwtheatres.co.uk/theatres/the-london-palladium/",
"https://www.academymusicgroup.com/o2academybrixton/",
"https://www.princeedwardtheatre.co.uk/",
```

```
"https://www.cromerpier.co.uk/",
"https://www.southbankcentre.co.uk/venues/royal-festival-hall",
"http://www.birminghamhippodrome.com/",
"https://lwtheatres.co.uk/theatres/theatre-royal-drury-lane/",
"https://www.nederlander.co.uk/dominion-theatre",
"https://palacetheatre.co.uk/",
"http://www.trch.co.uk/",
"https://lwtheatres.co.uk/theatres/adelphi",
"http://www.sondheimtheatre.co.uk/",
"https://www.nationaltheatre.org.uk/",
"https://www.princeofwalestheatre.co.uk/",
"http://www.roundhouse.org.uk/",
"https://www.mayflower.org.uk/",
"https://www.atgtickets.com/venues/palace-theatre-manchester/?utm_sourc
e=google&utm_medium=organic&utm_campaign=gmb",
"http://aldwychtheatre.com/",
"http://londoncoliseum.org/",
"https://trafalgartickets.com/cliffs-pavilion-southend/",
"http://www.shaftesburytheatre.com/"
]
```

websiteConfigs.json

```
[
  {
    "name": "www.shakespearesglobe.com",
    "url": "https://www.shakespearesglobe.com/",
    "selectors": {
      "eventCard": ".c-event-card",
      "title": ".c-event-card__title",
      "date": ".c-event-card__date",
      "location": ".c-event-card__venue",
      "link": ".c-event-card__link"
    }
  },
  {
    "name": "www.atgtickets.com",
    "url":
"https://www.atgtickets.com/venues/lyceum-theatre/?utm_source=google&ut
m_medium=organic&utm_campaign=gmb",
    "selectors": {
      "eventCard": ".WhatsOnPanel__ShowCardWrapper-sc-1q6jo28-0",
      "title": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3",
      "date": ".Text-sc-1pb273e-0.gwbjhH",
      "location": null,
```

```
      "link": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3 > a"
    }
  },
  {
    "name": "www.atgtickets.com",
    "url":
"https://www.atgtickets.com/venues/apollo-victoria-theatre/?utm_source=
google&utm_medium=organic&utm_campaign=gmb",
    "selectors": {
      "eventCard": ".WhatsOnPanel__ShowCardWrapper-sc-1q6jo28-0",
      "title": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3",
      "date": ".Text-sc-1pb273e-0.gwbjhH",
      "location": null,
      "link": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3 > a"
    }
  },
  {
    "name": "www.atgtickets.com",
    "url":
"https://www.atgtickets.com/venues/palace-theatre-manchester/?utm_sourc
e=google&utm_medium=organic&utm_campaign=gmb",
    "selectors": {
      "eventCard": ".WhatsOnPanel__ShowCardWrapper-sc-1q6jo28-0",
      "title": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3",
      "date": ".Text-sc-1pb273e-0.gwbjhH",
      "location": null,
      "link": ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3 > a"
    }
  }
]
```

potentialSelectors.json

```
{
    "eventCard": [
      ".c-event-card",
      ".WhatsOnPanel__ShowCardWrapper-sc-1q6jo28-0"
    ],
    "title": [
      ".c-event-card__title",
      ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3"
    ],
    "date": [
      ".c-event-card__date",
      ".Text-sc-1pb273e-0.gwbjhH"
```

```
    ],
    "location": [
      ".c-event-card__venue"
    ],
    "link": [
      ".c-event-card__link",
      ".WhatsOnPanel__ShowTitle-sc-1q6jo28-3 > a"
    ]
  }
```