README

PortalHSC

A student portal designed for tutoring centres.

Identifying and Defining

Define and analyse problem requirements

The following tables outlines both functional and performance requirements, prioritized from most essential to optional, distinguishing between core needs and potential opportunities for future implementation.

Needs

Feature	Functional Requirement	Performance Requirement
Login Functionality	The system will include a user authentication feaure that verifies user credentials before granting access to eprsonalised content	Login requets must be processed within 2 seconds under normal load
Solution Access	Students should be able to access questions located within a bank.	Should use a PDF viewer
Question Bank Access	Students should be able to access questions located within a bank.	Should use a PDF viewer
Admin Control	Admins should be able to manipulate databases	Be able to CRUD, create, read, update, deleted, SQL database tables.

Opportunities

Feature	Functional Requirement	Performance Requirement
Invoicing View	Students can access their invoice information, such as due dates and receipts	Dynamically linked to a SQL database with the invoices.

Scheduling and financial feasibility

Following comprehensive discussions with the client, the client HopeHSC requres a digital solution designed to streamline academic and adminstrative interactions between students and tutors. Therefore they have contacted us, the Fort Street Software Solutions Company, to create a tool that assists these students in preparation for the HSC. This tool will be a progressive web application (PWA) that increases efficiency of tasks such as marking homeworking, and managing invoices through digitisation. The name for this project will be "PortalHSC"

Functionality Requirements:

- Login Functionality for students and teachers to have accounts.
- Provide access to homework solutions.
- Offer a searchable question bank.

Performance Requirements:

- All core features must be fully functional offline.
- Maintain responsive performance and avoid lag
- Interoperability between devices of different manufacturers
- Avaliable through variety of internet browsers.
- All data retrieval and submission must be secure and reliable under varying network conditions.

All needs listed within the functional requirements will be completed before the optional oppurtunities. Prioritising these ensurs that the core user requirements are met, enabling basic tutoring and adminstrative processes, therefore suceeding in providing a solution for HopeHSC. The oppurtunities will be addressed later, as they enhance user experience but are not critical for the intial operation of the system.
All functionalities are independant of each other, except the interaction betwen allowing homeworks to submit homework online, and enabling tutors to provide feedback on this submitted homework.

Costs include:

- Development costs: Time and effort invested in designing, coding, testing and deploying the PWA.
- Hosting and Infrastructure: There may be a possibility of fees for cloud services or servers needed to host the application and store data securely

Entities, Data Structures, and Data Types

- Student
  - Data Structure: Array of records or database table
  - Data Types:
    - `String` – name, email

        - `Integer` – age, student ID
        - `Boolean` – active/inactive status
- Tutor
    - Data Structure: Array of records or database table
    - Data Types:
        - `String` – name, subject area
        - `Integer` – tutor ID
        - `Array` – list of assigned class IDs
- Homework
    - Data Structure: Array of records or database table
    - Data Types:
        - `String` – title, feedback comments
        - `Date` – due date
        - `File` – uploaded homework file
        - `Integer` – mark awarded
- Invoice
    - Data Structure: Array of records or database table
    - Data Types:
        - `Integer` – amount
        - `Date` – issue date, due date
        - `Boolean` – payment status
- Class
    - Data Structure: Array of records or database table
    - Data Types:
        - `String` – subject, location
        - `Date` – date and time
        - `Integer` – duration in minutes
        - `Array` – list of enrolled student IDs
- Solution
    - Data Structure: Array of records or database table
    - Data Types:
        - `String` – solution text
        - `File` – attachment
        - `Integer` – related homework ID
- Announcement
    - Data Structure: Array of records
    - Data Types:
        - `String` – title, content
        - `Date` – time posted

- Question Bank
  - Data Structure: Nested array or separate database table
  - Data Types:
    - `String` – question text, topic
    - `Integer` – difficulty level
    - `Boolean` – whether answered
- Gamification Badge (optional)
  - Data Structure: Array stored within student record
  - Data Types:
    - `String` – badge name
    - `Date` – date earned

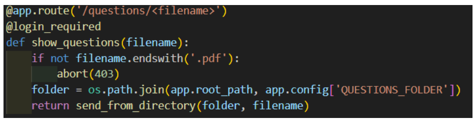
Boundaries

The tutoring portal PWA will operate within defined boundaries to ensure it remains manageable and be able to function its core purpose. Several aspects that fall under the services of HopeHSC will remain outside the system. These include:

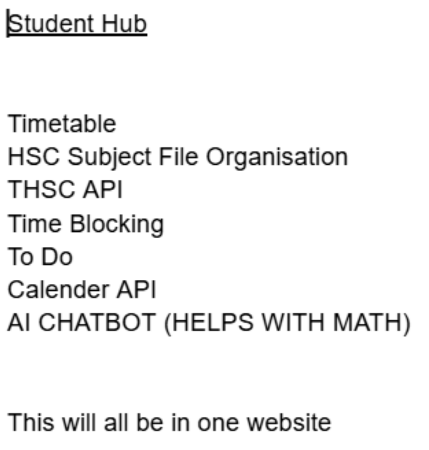
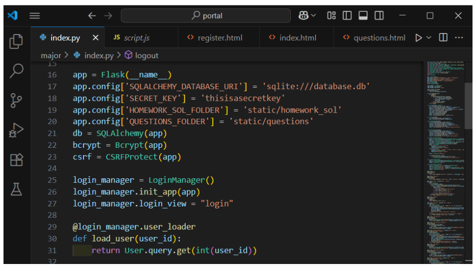
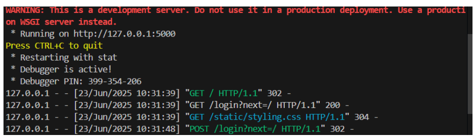
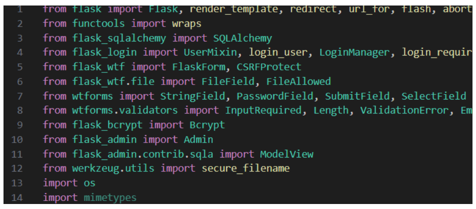
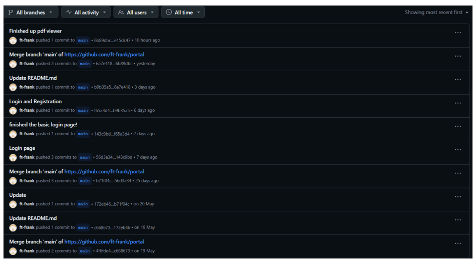
- Live tutoring sessions, which will occur using external platforms such as Zoom.
- Payment processing for invoices will not be handled directly; instead the system will only display invoices. Payments options will be decided by HopeHSC, however will likely involve either bank transfer and cash payments in termly instalments.
- Authentication systems such as Google Oauth aren't necessary, as there is limited to no malicious activity possible and the userbase will be private. Accounts will be created as required by admins.
- Any policies, curriculum changes and rules will be agreed to and signed externally.
- Homework will be provided physically by the tutoring centre, as they wish to gatekeep their resources and keep them safe.

Tools

A description of a variety of different tools used during the production and implementation of the PWA.

Tool	Screenshot	Description
Algorithm Design		One of the most difficult features of this project was ensuring that the database of files would function properly. This algorithm was the hardest to execute properly, and thus hours of algorithm design resulted in this snippet of code.

```python
@app.route('/questions/<filename>')
@login_required
def show_questions(filename):
    if not filename.endswith('.pdf'):
        abort(403)
    folder = os.path.join(app.root_path, app.config['QUESTIONS_FOLDER'])
    return send_from_directory(folder, filename)
```

Tool	Screenshot	Description
Brainstorming	 Student Hub Timetable HSC Subject File Organisation THSC API Time Blocking To Do Calender API AI CHATBOT (HELPS WITH MATH) This will all be in one website	Before our client changed project requirements, months before working on the project around February our engineers had developed a plan for the then planned Student Hub
Code Generation		Visual Studio Code is the code editor that allows this project's code to be generated and compiled.
Data Dictionaries	Found in Research and Planning	
Debugging		Within the terminal within Visual Studio Code, as the index.py has debug=True, it allows the developer to easily identify the root of bugs
Installation		The project has required many modules such as flask-admin and flask_sqlalchemy to function, so these have been installed using the pip package installer.
Maintenance		Using Github, over the course of the development of the project, our developer has maintained updates and fixes to the code of the project.
Storyboards	Found in Research and Planning -	
Testing	Found in Testing and Evaluating -	

```python
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'thisisasecretkey'
app.config['HOMEWORK_SOL_FOLDER'] = 'static/homework_sol'
app.config['QUESTIONS_FOLDER'] = 'static/questions'
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
csrf = CSRFProtect(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

```
WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 399-354-206
127.0.0.1 - - [23/Jun/2025 10:31:39] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [23/Jun/2025 10:31:39] "GET /login?next=/ HTTP/1.1" 200 -
127.0.0.1 - - [23/Jun/2025 10:31:39] "GET /static/styling.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Jun/2025 10:31:48] "POST /login?next=/ HTTP/1.1" 302 -
```

```python
from flask import Flask, render_template, redirect, url_for, flash, abort
from functools import wraps
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_requir
from flask_wtf import FlaskForm, CSRFProtect
from flask_wtf.file import FileField, FileAllowed
from wtforms import StringField, PasswordField, SubmitField, SelectField
from wtforms.validators import InputRequired, Length, ValidationError, Em
from flask_bcrypt import Bcrypt
from flask_admin import Admin
from flask_admin.contrib.sqla import ModelView
from werkzeug.utils import secure_filename
import os
import mimetypes
```

Finished up pdf viewer
ft-frank pushed 1 commit to main • 6b89dbc...e15dc47 • 10 hours ago

Merge branch 'main' of https://github.com/ft-frank/portal
ft-frank pushed 2 commits to main • 6a7e418...6b89dbc • yesterday

Update README.md
ft-frank pushed 1 commit to main • b9b35a5...6a7e418 • 3 days ago

Login and Registration
ft-frank pushed 1 commit to main • f65a3d4...b9b35a5 • 6 days ago

finished the basic login page!
ft-frank pushed 1 commit to main • 143c9bd...f65a3d4 • 7 days ago

Login page
ft-frank pushed 3 commits to main • 56d3a34...143c9bd • 7 days ago

Merge branch 'main' of https://github.com/ft-frank/portal
ft-frank pushed 3 commits to main • b71f04c...56d3a34 • 25 days ago

Update
ft-frank pushed 1 commit to main • 172eb46...b71f04c • on 20 May

Update README.md
ft-frank pushed 1 commit to main • c668673...172eb46 • on 19 May

Merge branch 'main' of https://github.com/ft-frank/portal
ft-frank pushed 2 commits to main • 4f6fde4...c668673 • on 19 May

Software implementation methods.

Pilot implementation involves rolling out the new system to a small, manageable group of users before a full-scale implementation. This method allows organisations to identify any issues or necessary adjustments in a controlled environment, reducing the risk of widespread problems.

A pilot implementation is beneficial as by letting the tutoring portal be tested by a small group first, they can help identify and fix issues without wasting other students' time potentially dealing with a flawed system. Once the group and developer are satisfied with the solution, then it may be distributed amongst the student body as a complete package, either immediately replacing the old system or filling in the digital gap within HopeHSC.

Direct implementation would be risky, as the new system has not been thoroughly tested.
Parallel implementation would confuse the administrative, student and teaching staff during operations.
Phased implementation will take too long to implement.

Research and Planning

Project Management

Project management is the process of planning, organizing, and overseeing tasks and resources to achieve specific project goals within a set timeframe and budget. It ensures that a project is completed efficiently, meets requirements, and delivers value to clients

Software Development Approaches

<u>The Waterfall Software Development Approach</u>

Question	Sample Explanation
1.1 How are the logical progression of steps used throughout the life cycle?	The Waterfall model follows a strict, linear sequence of stages. Each stage must be fully completed before the next begins, ensuring a clear and logical flow.
1.2 What are the stages of 'falling water'?	Requirements Gathering System Design Implementation (Coding) Testing Deployment Maintenance
1.3 What are the advantages and disadvantages of this approach?	Advantages: - Easy to manage due to its rigid structure - Good for small or well-defined projects - Documentation is thorough and complete Disadvantages: - Inflexible to changes - No working product until late in the process

Question	Sample Explanation
	- Late discovery of issues during development can be costly
1.4 Give examples of the scale and types of developments that use this approach.	- Large-scale government or defence systems (tax systems) - Large construction and infrastructure (bridges, air traffic control) - Large-scale manufacturing projects (cars, vehicles) - Large-scale healthcare projects (medical-record systems, medicinal rollout)

The WAgile Software Development Approach

Question	Sample Explanation
2.1 Explain why it is a hybrid model	WAgile is hybrid as it combines the structure of Waterfall (upfront planning, documentation, etc.) and Agile's flexibility (e.g., iterations, user feedback).
2.2 Analyse the 'when' intervention is applied during the development life cycle	Agile practices (like stand-ups, iterative development, testing) are introduced after the initial Waterfall stages, often during implementation or testing. It may start rigid but loosen control during later phases for adaptability.
2.3 Analyse the 'how' intervention is applied during the development life cycle	Agile interventions are layered into Waterfall by: - Splitting implementation into sprints - Including regular stakeholder reviews - Allowing feedback loops during testing This hybridization enables flexibility while maintaining upfront planning.
2.4 Give examples scale and types of developments that use this approach	- Medium to large projects in corporate environments - Government or healthcare systems with compliance requirements - Projects with fixed deadlines but evolving features

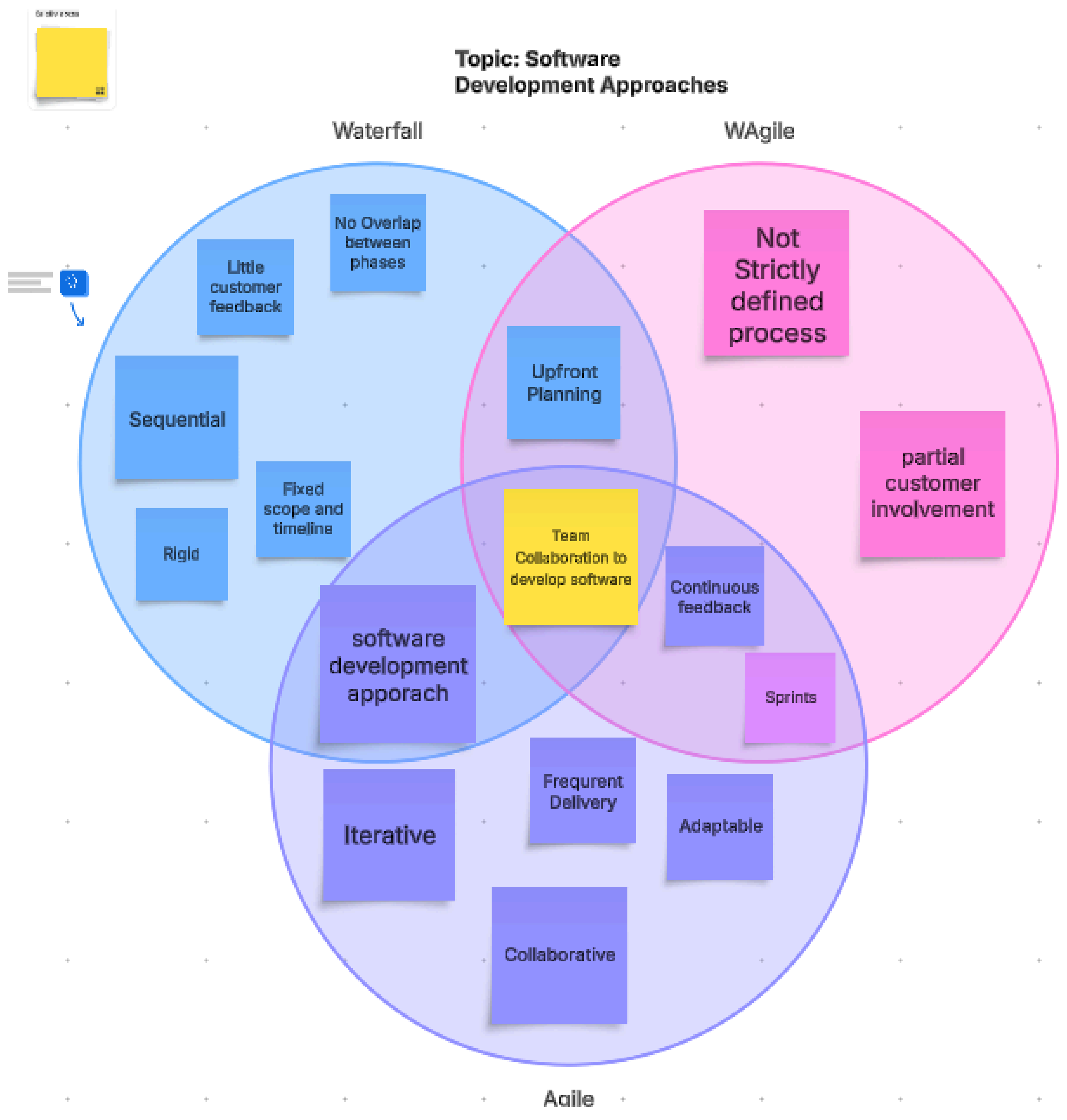
---

The Agile Software Development Approach

Question	Sample Explanation
3.1 What is the rate of developing a final solution?	Agile delivers a working product early and often, typically every 1-4 weeks in sprints.

Question	**Sample Explanation**
3.2 Explain method tailoring	Method tailoring involves adapting Agile methods (like Scrum, Kanban) to suit the team or project. For example, adjusting sprint lengths, roles, or tools to match the team’s needs and the project scope.
3.3 Explain iteration workflow	Each iteration (or sprint) concludes a round of: - Planning - Design - Development - Testing - Review After each cycle, feedback is incorporated into the next iteration, enabling rapid improvements.
3.4 Give examples of the scale and types of developments that use this approach	- Web and mobile app startups - SaaS platforms - Games and creative media projects - Generally all small to medium sized projects

Venn Diagram


Topic: Software Development Approaches
Waterfall
WAgile
Agile
No Overlap between phases
Little customer feedback
Sequential
Fixed scope and timeline
Rigid
Upfront Planning
Not Strictly defined process
partial customer involvement
Team Collaboration to develop software
Continuous feedback
Sprints
software development apporach
Iterative
Frequrent Delivery
Adaptable
Collaborative

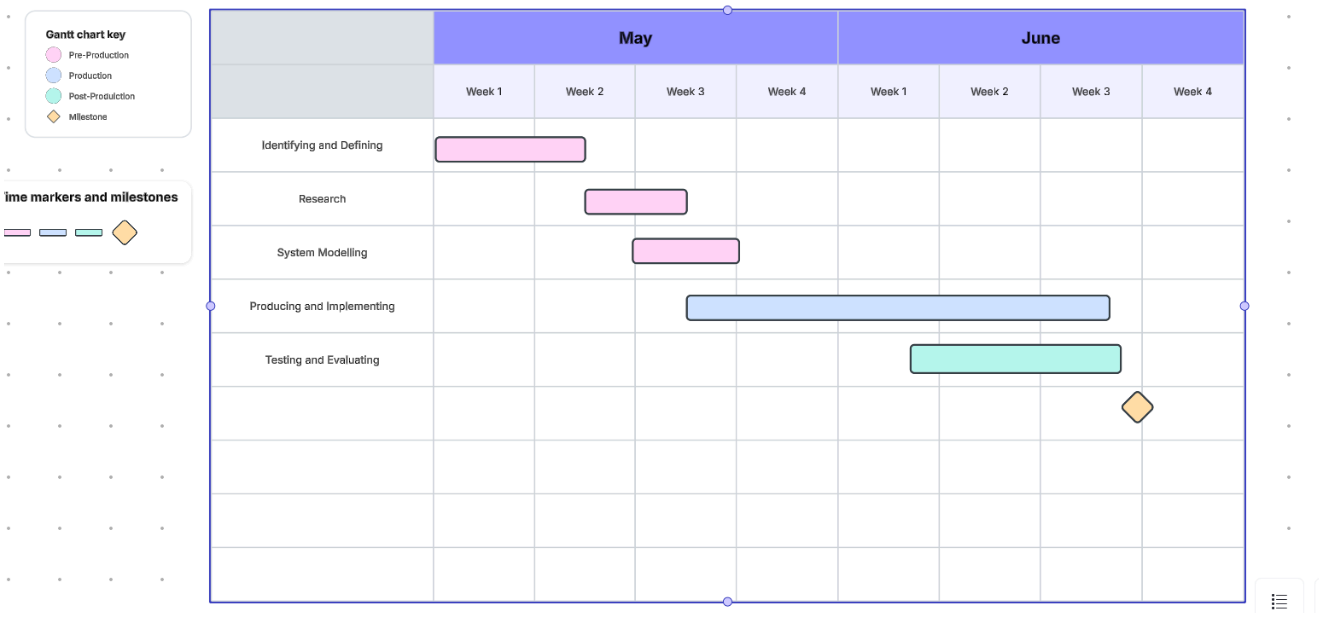
<u>Development Approach</u>

A **Wagile** approach to development would be implemented for PortalHSC due to the its balance of structure, planning and flexibility in time-restrained projects.

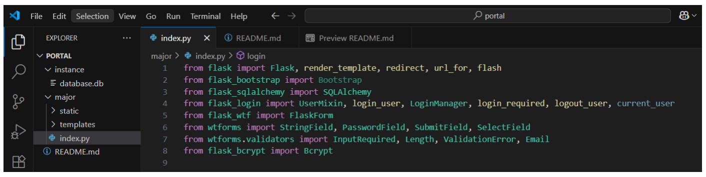
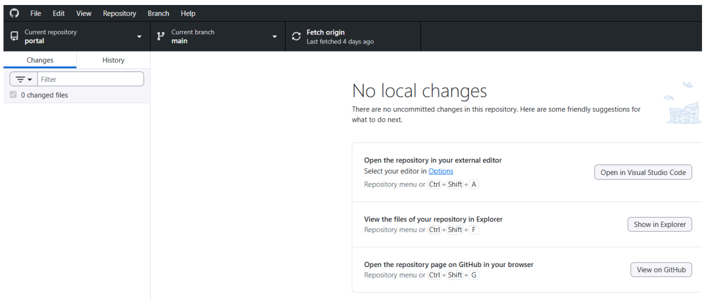
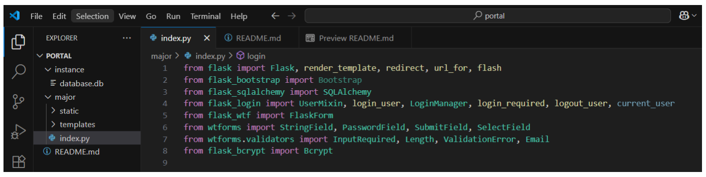
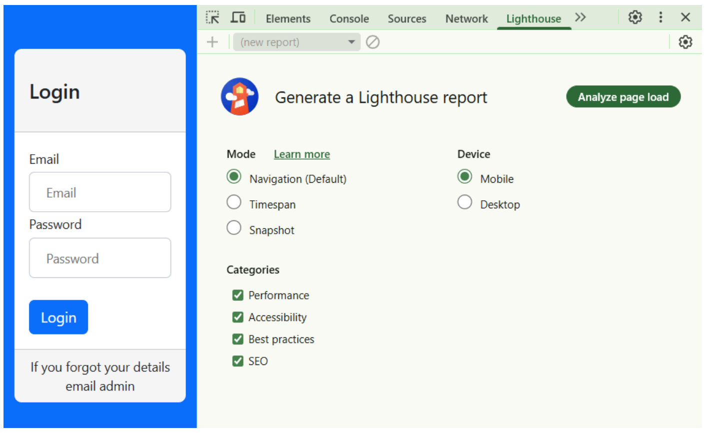
The WAgile development process is the most efficient choice for PortalHSC project, as it combines the structured planning of Waterfall with the flexibility of Agile. This hybrid model allows for clear documentation and requirement definition, which is ideal for meeting the client's standards (school project) whilst also supporting iterative devleopment and feedback during implemenation. WAgile ensures the project stays organised yet adaptable, making it well-studied for our solo developer team with fixed deadlines and evolving feature needs.

Scheduling and Task allocation

Software Major Project "Portal(HSC)"



Collaboration Tools

Tool	Description	Screenshot
Visual Studio Code	A lightweight code editor used for writing and managing source code.	
GitHub	A platform for version control and collaboration using Git.	
Flask	A Python web framework used to develop the PWA backend.	
Chrome DevTools	Browser tools for debugging and testing web applications.	

```
major > index.py > login
from flask import Flask, render_template, redirect, url_for, flash
from flask_bootstrap import Bootstrap
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, SelectField
from wtforms.validators import InputRequired, Length, ValidationError, Email
from flask_bcrypt import Bcrypt
```

Current repository portal · Current branch main · Fetch origin Last fetched 4 days ago

0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl + Shift + A — Open in Visual Studio Code

View the files of your repository in Explorer
Repository menu or Ctrl + Shift + F — Show in Explorer

Open the repository page on GitHub in your browser
Repository menu or Ctrl + Shift + G — View on GitHub

```
major > index.py > login
from flask import Flask, render_template, redirect, url_for, flash
from flask_bootstrap import Bootstrap
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, SelectField
from wtforms.validators import InputRequired, Length, ValidationError, Email
from flask_bcrypt import Bcrypt
```

Login

Email: Email

Password: Password

Login

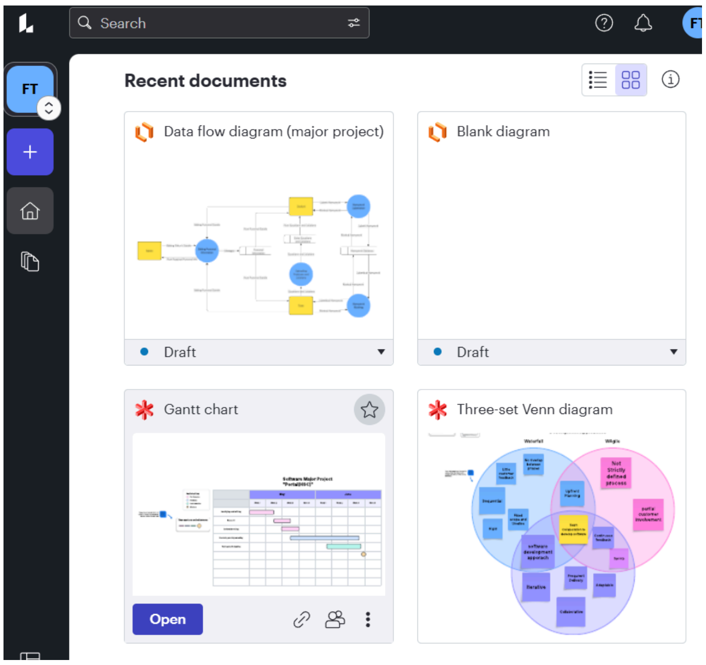
If you forgot your details email admin

Generate a Lighthouse report — Analyze page load

Mode: [✓ Navigation (Default)] Timespan Snapshot

Device: [✓ Mobile] Desktop

Categories: ✓ Performance ✓ Accessibility ✓ Best practices ✓ SEO

Tool	Description	Screenshot
Lucidchart	A diagramming tool used to plan system architecture and workflows.	 Search FT Recent documents Data flow diagram (major project) Blank diagram Draft Draft Gantt chart Three-set Venn diagram Open

Social and Ethic aspects of software enginering projects

<u>Privacy</u>

Data required by the client from each student will involve:
-Student Name
-School
-Student Email

Privacy is essential when handling student information such as names, emails. All data will be stored securely and not shared with third parties.

<u>Security</u>

The issue of security applies to the process of homework submissions and account login. To prevent common threats like cross-site scripting (XSS) and SQL injection, all inputs will be validated and sanitised. Additionally, passwords will be hashed and HTTPS will be used to encrypt data in transit. Homework submission will also be limited to PDF.

<u>Accessibility and Inclusivity</u>
Accessibility ensures all students can use the portal regardless of ability. The UI will follow WCAG guidelines by using proper colour contrast, alt test for images, and keyboard navigation.

<u>Transparency</u>
Transparency means users should know how the platform works and what it does with their data. A clear “How it works” section will explain features such as homework tracking, feedback systems, and how logins are handled.

Fairness

Fairness ensures that our software does not discriminate against any group or individual. Features such as homework feedback, class details and resource access etc. must be equally available to all users regardless of their background and avoid biases in its functionality.

Intellectual Property
The platform will respect intellectual property laws by using only open-source assets. Any third-party frameworks and libraries used(e.g. Bootstrap, Flask) will be attributed and licensed. Tutors uploading content will agree not to upload copyrighted material without permission.

Collaboration
Collaboration between the client, our developers and the clients' students is essential in building a successful solution. Regular feedback from tutors during development would help align features with real needs, such as homework return systems.

Feedback
PortalHSC will include a feedback form that allows students and tutors to report bugs or suggest improvements. This feedback will be reviewed regularly, and the information gathered will be used to plan future updates. Feedback encourages a user-focused process of development and ensures the platform continues to improve.

Quality Assurance

Quality Criteria	**Explanation**
Google Lighthouse	Google Lighthouse is an automated tool developed by Google that analyses web pages and generates a score from 0 to 100 based on various factors like performance, accessibility, best practices, SEO and PWA. An ideal score is anywhere from 90-100.
Response Timme	The software should respond to all user requests in a time of less than 3 seconds
Interoperability	Testing on multiple web browsers

Quality Criteria	**Description**
Google Lighthouse	Google Lighthouse is an automated tool developed by Google that analyses web pages and generates a score from 0 to 100 based on various factors like performance, accessibility, best practices, SEO and PWA. An ideal score is anywhere from 90-100.
Responsiveness	The interface should work seamlessly on mobile, tablet, and desktop devices.

Quality Criteria	Description
Accessibility Compliance	Meets WCAG 2.1 Level AA guidelines for accessibility.
Ease of Navigation	Users can access any feature within 2–3 clicks from the homepage.
Minimal Load Time	Pages should load in under 2 seconds on a standard connection.

Compliance and Legislative Requirements

Compliance or Legislative Issue	Methods for Mitigation
Privacy Act 1988 (Cth)	Collect only necessary user data, display a clear privacy policy, obtain user consent before data collection, and ensure data is securely stored and not shared without permission.
Privacy and Personal Information Protection Act 1998 (NSW)	Implement access controls, limit data visibility to relevant users (e.g., tutors can only see their students), and encrypt sensitive information.
Spam Act 2003 (Cth)	Ensure that any communications (e.g., reminders or announcements) include consent and an option to opt out.
Copyright Act 1968 (Cth)	Use only licensed or original materials (e.g., icons, past paper content), and credit sources when required.
Australian Consumer Law	Provide accurate, non-misleading descriptions of the PWA's features, especially if a payment component (e.g., invoicing) is integrated.
ISO/IEC 27001 (Information Security Management)	Follow industry best practices for information security: implement strong password rules, use HTTPS, and conduct regular security audits.

System Modelling

Data Dictionary

User Table

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	V
id	Integer	1	4	Integer	Unique user ID		A g

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	V
					(Primary Key)		p k
email	String	admin@gmail.com	100	Text	Email address of the user		F U V e f
password	String	admin	100	Hidden	Hashed user password		F M c
class_id	Integer	Ext 2	4	Integer	Associated class ID (Foreign Key to Class)		C (n
role		Admin	20	Text	Role of user		F M c a t s
first_name	String	Frank	50	Text	First name of the user		C
last_name	String	Tran	50	Text	Last name of the user		C
school	String	Fort Street High School	100	Text	School the student attends		C
grade	String	12	20	Text	Student grade level		C

Class Table

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
id	Integer		4	Integer	Unique class ID (Primary Key)		Auto-generated

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
name	String		50	Text	Name of the class		Required

Homework Sol Table

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
id	Integer		4	Integer	Unique ID for homework solution		Auto-generated
title	String		100	Text	Title of homework		Required
filename	String		100	Text	File name of uploaded document		Required, ends with .pdf/.docx
class_id	String		100	Text	Related class (referenced by ID or name)		Required
file_path	String		200	File path	Path to where file is stored		Required

QuestionBank Table

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
id	Integer		4	Integer	Unique question set ID		Auto-generated
title	String		100	Text	Title of the question paper		Required
filename	String		100	Text	Name of uploaded		Required, ends with

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
					question file		.pdf/.docx
class_id	String		100	Text	Related class (referenced by ID or name)		Required
file_path	String		200	File path	Path to the stored file		Required

Flask variables and functions

Variable	Data Type	Format for Display	Description	Validation
`app`	Flask Object	App Instance	Main Flask app instance that runs the web server	Initialized once and configured with settings
`db`	SQLAlchemy	DB Object	Database handler using SQLAlchemy ORM	Configured with a valid URI
`bcrypt`	Bcrypt	Encryption Handler	Used to hash and verify user passwords securely	Instantiated once for app
`csrf`	CSRFProtect	Security Middleware	Protects Flask app against CSRF attacks	Attached to app once
`login_manager`	LoginManager	Auth Handler	Handles user session and login behavior	Must define login view and user_loader
`Admin`	Flask-Admin	Admin Dashboard	Provides an admin dashboard to manage database	Must be registered with views and secured access
`secure_filename`	Function	Utility Function	Sanitizes uploaded filenames for safe storage	Must be used on user-uploaded filenames

Variable	Data Type	Format for Display	Description	Validation
`wraps`	Function Decorator	Decorator	Preserves function metadata in custom decorators	Used when writing role-based decorators

Session Variables Data Dictionary

Variable	Data Type	Format for Display	Size (Bytes)	Size for Display	Description
`current_user.id`	Integer	User ID	4	Integer	ID of the currently logged-in user
`current_user.email`	String	Email	~50	Text (email)	Email of the authenticated user
`current_user.role`	String	Role	~20	Text (enum)	Role of the user (`student` , `tutor` , `admin`)
`current_user.class_id`	Integer or None	Class ID	4	Integer or null	Foreign key to user's assigned class (if any)
`login_user()`	Function	-	N/A	N/A	Logs in a user and stores session state
`logout_user()`	Function	-	N/A	N/A	Clears the current user from session
`login_manager.login_view`	String	Endpoint name	~30	URL string	Specifies the default route for unauthorized access

Register Form

Field Name	**Data Type**	**Format for Display**	**Size (Bytes)**	**Size for Display**	**Description**	**Example**	**Valida**
email	String	Email	~50	Text (email)	Email address of new user		Requi valid email format unique
password	String	Password	~100 (hash)	Text (password)	Password for user account		Requi 4–20 charac
role	String	Select (`student`, `tutor`, `admin`)	~20	Text (enum)	Role assigned to user		Requi must b one of allowe choice
submit	Submit	Button	-	-	Form submission button		-

Login Form

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validatior
email	String	Email	~50	Text (email)	User email for login		Required
password	String	Password	~100	Text (password)	User password for login		Required
submit	Submit	Button	-	-	Form submission button		-

Homework Form

Field Name	Data Type	Format for Display	Size (Bytes)	Size for Display	Description	Example	Validation
title	String		~100	Text	Title of homework or question		Required
file	File		Varies	File	File to upload		Required, filetype allowed
class_id	Integer	Extension 2	4	Integer	ID of class assigned to upload		Required, must match valid Class ID
submit	Submit	y	-	-	Form submission button		-

/Home

Variable	Type	Format	Description	Example	Validation
current_user	Object	Authenticated User	Logged-in user	-	Must be authenticated

/Login

Variable	Type	Format	Description	Example	Validation
form	LoginForm	WTForms	Handles login credentials	-	Must be valid, correct user/pass

/logout

Variable	Type	Format	Description	Example	Validation
form	RegisterForm	WTForms	Handles account creation	-	Email must be unique, valid

/invoices

Variable	Type	Format	Description	Example	Validation
current_user	Object	User	Retrieves invoice page	-	Must be authenticated

/homework

Variable	Type	Format	Description	Example	Validation
homeworks	List	List of HomeworkSol	Displays all uploaded homework solutions	-	Auth required

/settings

Variable	Type	Format	Description	Example	Validation
current_user	Object	User	Access settings	-	Auth required

/class

Variable	Type	Format	Description	Example	Validation
classes	List	List of Class objects	Displays all classes	-	Auth required

/questions

Variable	Type	Format	Description	Example	Validation
questions	List	List of QuestionBank	Displays uploaded questions	-	Auth required

`/questions/<filename>`

Variable	Type	Format	Description	Example	Validation
filename	String	Path param	Downloads/displays question file	-	Must be PDF and exist

`/upload_questions`

Variable	Type	Format	Description	Example	Validation
form	HomeworkForm	WTForms	Uploads a new question to question bank	-	File, title, and class required

`/upload_homework_sol`

Variable	Type	Format	Description	Example	Validation
form	HomeworkForm	WTForms	Uploads a new homework solution	-	File, title, and class required

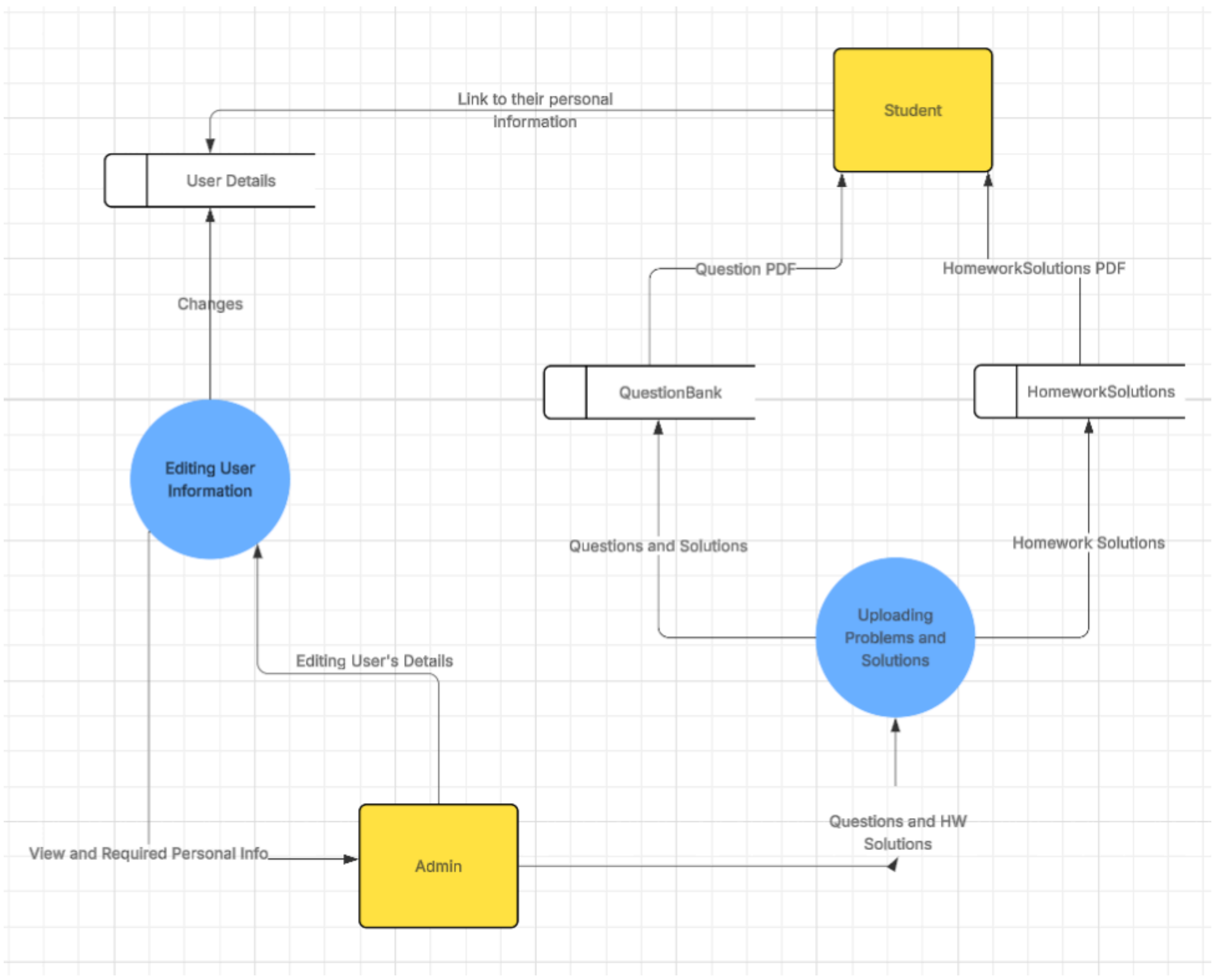
`/hw_solutions`

Variable	Type	Format	Description	Example	Validation
homeworks	List	List of HomeworkSol	Displays all uploaded homework solutions	-	Auth required

`/hw_solutions/<filename>`

Variable	Type	Format	Description	Example	Validation
filename	String	Path param	Opens or downloads specific solution	-	Must end with .pdf

Dataflow Diagram



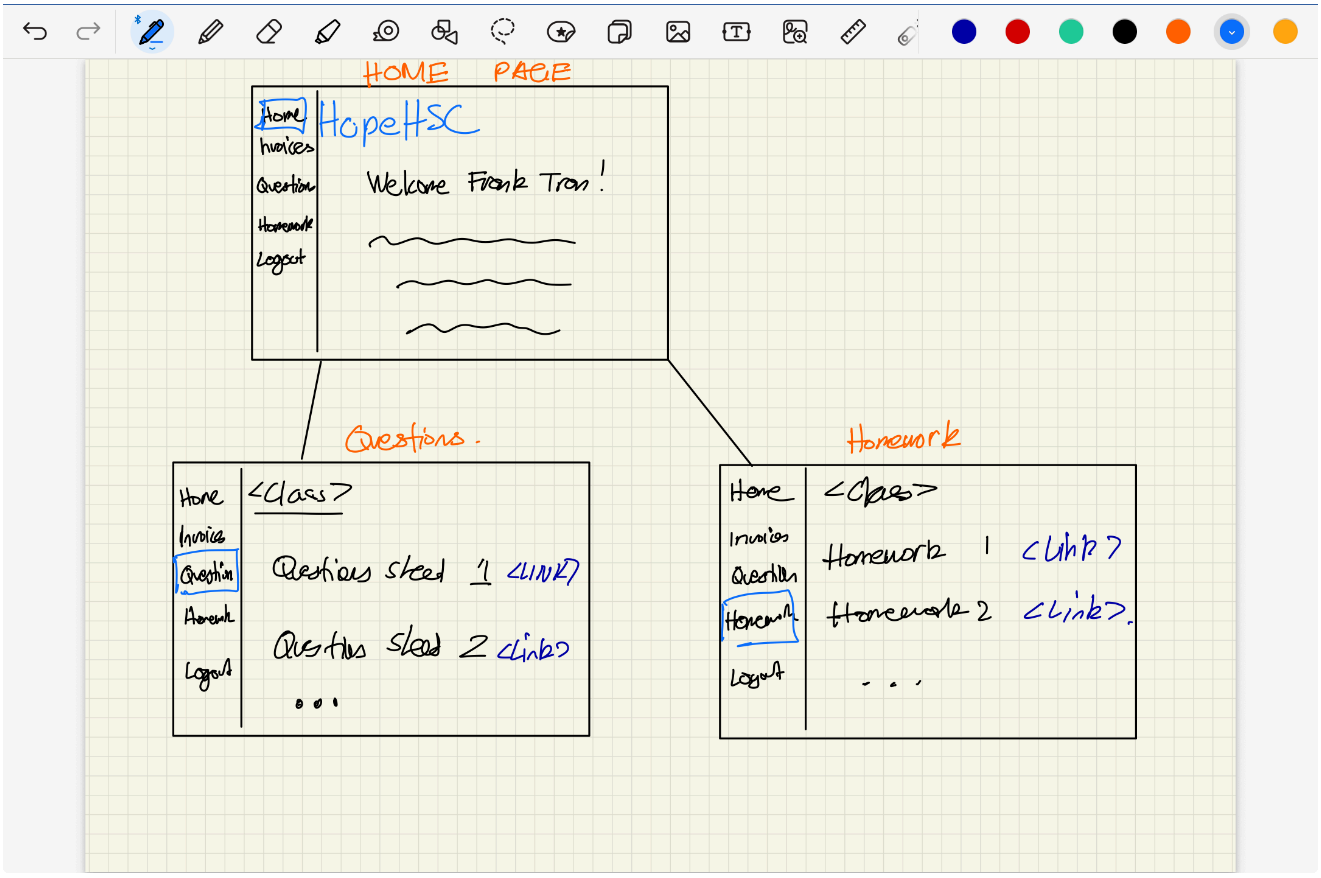
Structure Chart

https://lucid.app/lucidchart/3f6ae1df-2a89-4ef7-9724-a8e04396ea32/edit?viewport_loc=2831%2C388%2C3630%2C1998%2C0_0&invitationId=inv_cf40cc99-4248-455c-9de2-c9cfeed24f75

Class Diagram

https://lucid.app/lucidchart/7d0679fa-4f30-4f94-821b-cd95ddc042d5/edit?viewport_loc=-1587%2C-1088%2C5586%2C2631%2C0_0&invitationId=inv_848fa86c-5b52-4bb3-a2ea-fdcccf41d94d

Storyboard



Decision Tree
https://lucid.app/lucidchart/baee6e08-7e22-4efb-876c-b7992b26efc3/edit?invitationId=inv_0a30ec1c-c668-41bc-9fa5-5a52585aa1f5

Algorithm Design

Pseudocode for the algorithm for displaying a PDF file.

```
BEGIN ShowPDF(filename)

    IF filename does NOT end with ".pdf" THEN

        ABORT with error 403

    ENDIF



    DECODE filename from URL (replace %20 and other unsafe chars)

    SET folder = static/questions or static/homework_sol

    RETURN file using send_from_directory with appropriate mimetype
```

```
END
```

Backend Engineering

Terminology	Technology Used in This Project
Backend Engineering	Flask (Python), SQLite (relational database), SQLAlchemy (ORM)
Error Handling	Flask `abort()`, `try/except` blocks, WTForms validation, Flash messages
Interfacing with Frontend	Flask routes (`@app.route()`), Jinja2 templating, `url_for()` for dynamic routing
Security Engineering	Flask-Login, Flask-WTF (CSRF protection), bcrypt (password hashing), RBAC

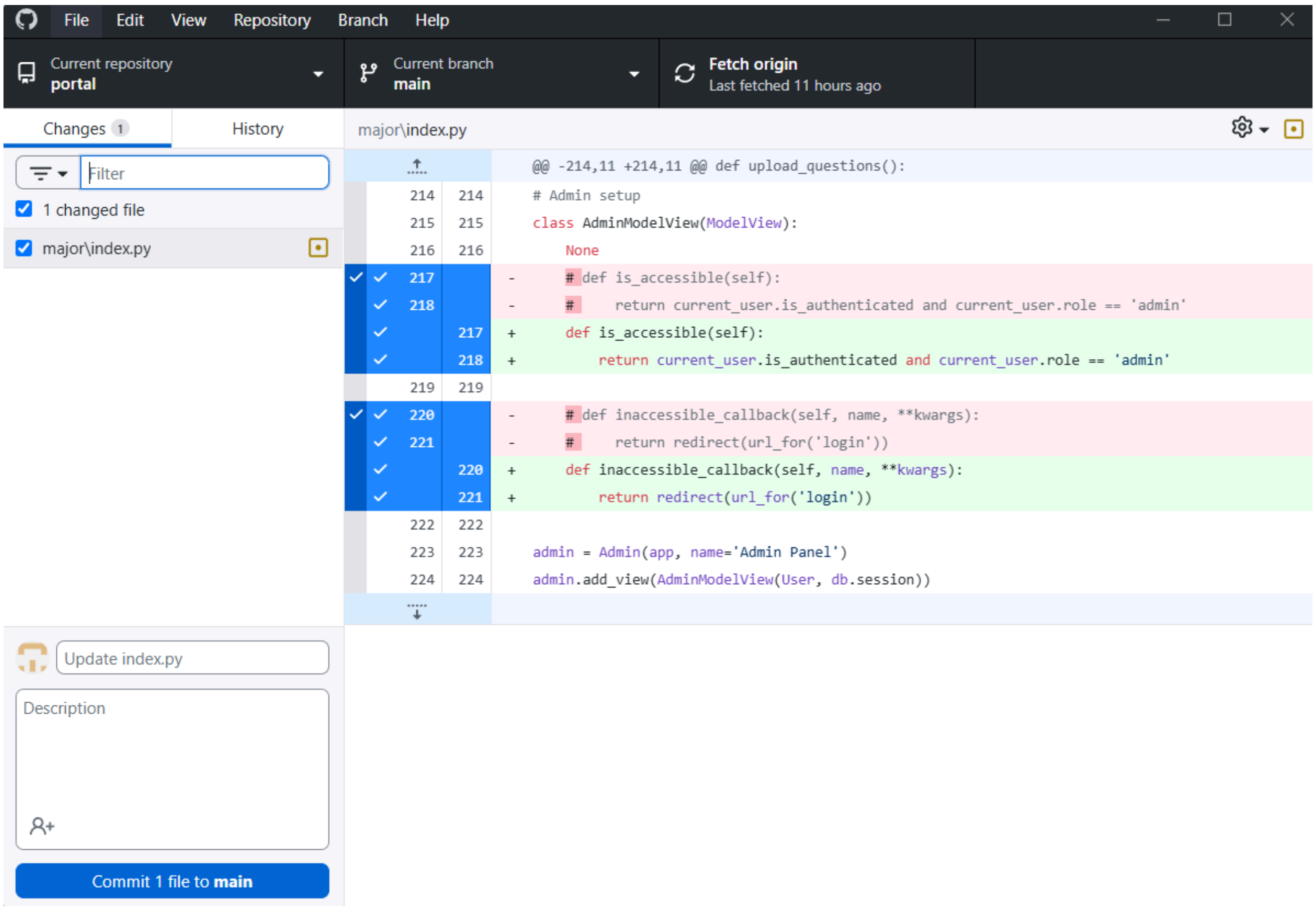
Producing and implementing

Documentation

Github.

Version Control

As our developer is relatively new to Github and Git, he has not fully grasped an understanding of using git bash, the terminal and github together. However there is no fear!, as Github Desktop is here!. Github Desktop was a simple, efficient, and innovative solution to version control, and has resulted in many successful debugging attempts.



major\index.py

```
@@ -214,11 +214,11 @@ def upload_questions():
    # Admin setup
    class AdminModelView(ModelView):
        None
-       # def is_accessible(self):
-       #     return current_user.is_authenticated and current_user.role == 'admin'
+       def is_accessible(self):
+           return current_user.is_authenticated and current_user.role == 'admin'

-       # def inaccessible_callback(self, name, **kwargs):
-       #     return redirect(url_for('login'))
+       def inaccessible_callback(self, name, **kwargs):
+           return redirect(url_for('login'))

    admin = Admin(app, name='Admin Panel')
    admin.add_view(AdminModelView(User, db.session))
```

Testing and Evaluating

<u>Optimisation Techniques</u>

Common language-dependent code optimisation techniques	**Description**
Python Optimisations	- Use of list comprehensions and generator expressions for memory-efficient loops. - Use of `@login_required` and `@role_required` decorators to avoid repeating access checks. - Query optimisation with SQLAlchemy (e.g. `.all()` instead of fetching per call). - Avoid global variables, use Flask's `current_user` for session state. - Lazy loading large files only when needed (e.g., PDFs via routes).
Javascript	- Only minimal DOM manipulation via event listeners. - Use of `defer` in `<script>` tags to delay execution until HTML parsing is complete. - Reuse DOM references to reduce lookups. - Avoid memory leaks by removing unused listeners.
HTML	- Semantic HTML tags used for accessibility and faster browser parsing.

Common language-dependent code optimisation techniques	Description
	- Template inheritance with Jinja2 (`{% extends "layout.html" %}`) to avoid repetition. - Minimize inline scripts/styles. - Lazy loading of external content (e.g. using `target="_blank"` for large files).
CSS	- Reuse Bootstrap classes instead of writing new styles. - Group similar selectors and avoid deeply nested rules. - Move custom styles to a separate `.css` file for caching. - Avoid unused styles (CSS bloat) and minimize redundancy.

Strategies during debugging

Version Control - In attempts to add new features, new functions, new databases etc. The code would run into issues relating to conflicting code. Using Github, our developers could retrace our steps and identify exactly where the code broke, and reverse engineer a solution.

Stack Overflow - Stack Overflow is a forum filled to the brim with all coders from talented to gifted to newbie. Viewing previous questions asked years ago that relate to my bug was one of the greatest tools in circumventing bugs.

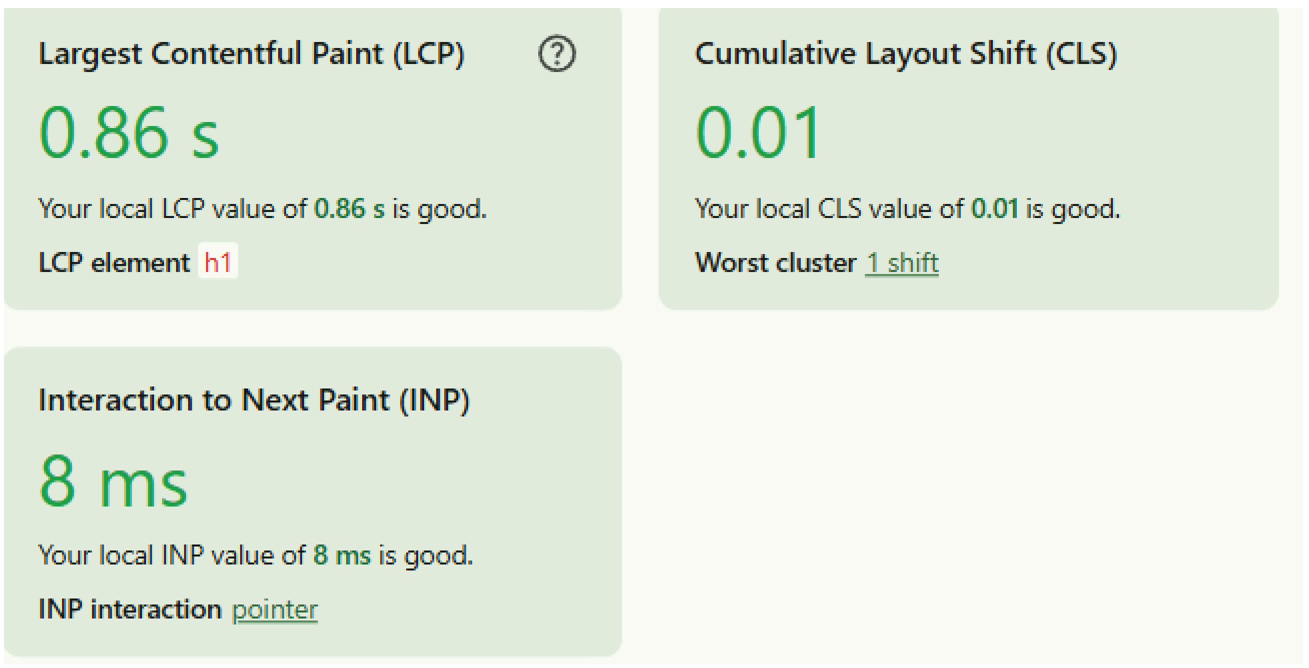
Documentation - Every module and framework with any decency would have accompanying documentation that would often outline all functions and capabilities. This documentation would allow our developers to ease into learning new modules for this project.

Debug Mode - Flask Debug Mode would sometimes pinpoint the line of code that would cause the program to break, thus saving our developers countless hours of time.

Success Criteria

Quality criteria	Met?	Analysis
Accessibility	Yes	Google Lighthouse gives a consistent score of above 98 for the accessibility of each of the webpages.
Responsiveness	Yes	Yes, the local INP value if always below 15ms, thus google has stated it as 'good'.
Interoperability	Yes	Works on: Firefox, Edge, Google Chrome. Therefore the interoperability has already reached more than 90% of people. Further testing is to be done.

Testing



Largest Contentful Paint (LCP)

0.86 s

Your local LCP value of 0.86 s is good.

LCP element h1

Cumulative Layout Shift (CLS)

0.01

Your local CLS value of 0.01 is good.

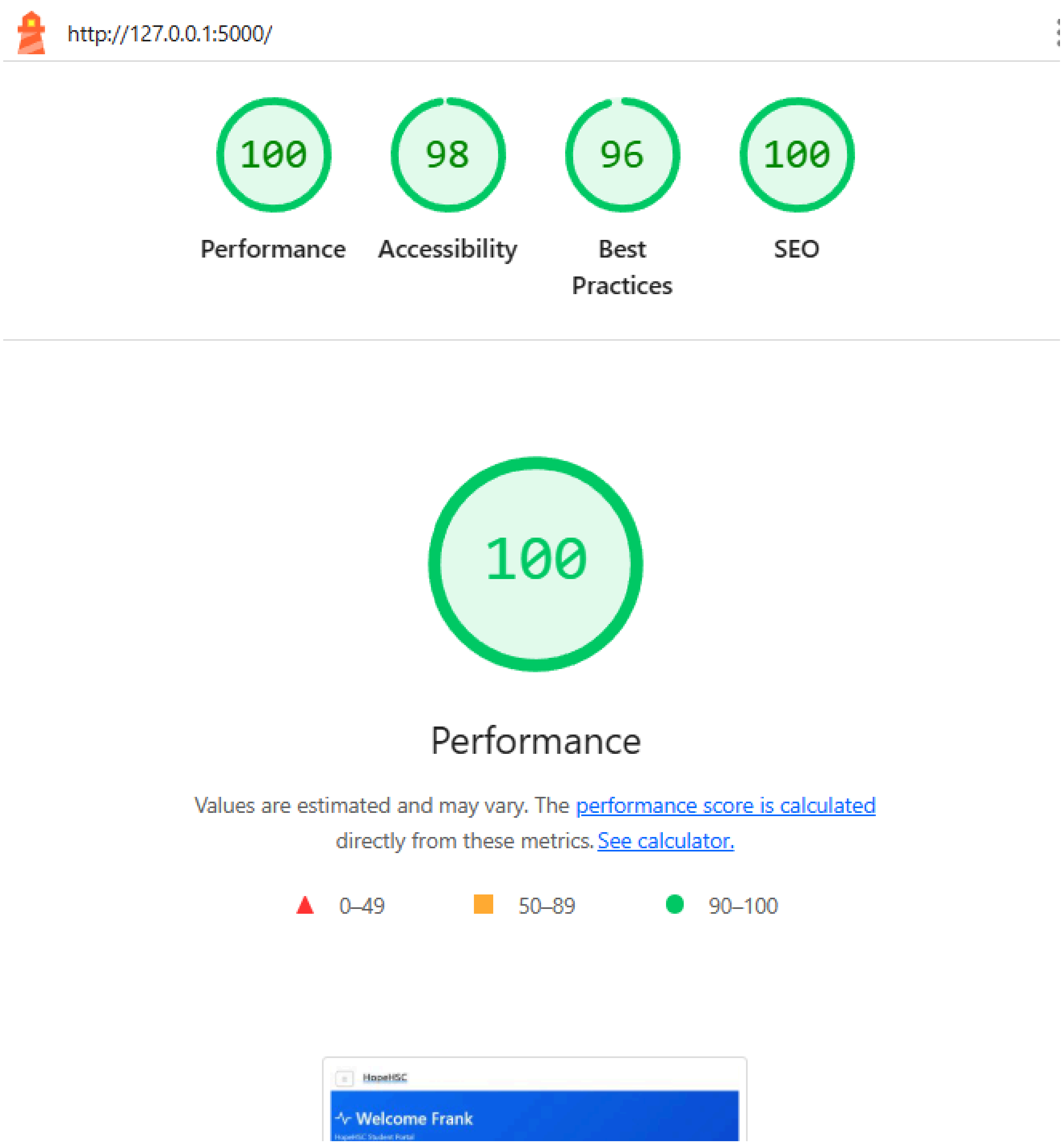
Worst cluster 1 shift

Interaction to Next Paint (INP)

8 ms

Your local INP value of 8 ms is good.

INP interaction pointer



http://127.0.0.1:5000/

100	98	96	100
Performance	Accessibility	Best Practices	SEO

100

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

**

Developing a report to synthesise feedback

- usability
  - We conducted multiple rounds of testing through verbal feedback, and they reported that
  - Navigation was clear, clean and intuitive.
  - However some had found that the primary blue theme was a bit 'jarring'.
- performance conclusions
  - Our testers had no complaints of the loading speeds of the application, and use on smaller screen sizes also had no problems
  - One picky tester did complain that the size of the texts was too large, and they suggested a more smaller UI.
- recommendations.
  - Our testers had many recommendations. These included suggestions of colour and theme change with a more monotone colour theme.
  - Tutors recommended features such as searching by class or student name in the homework view for easier navigation.
  - Some testers noted that while the app is functional on mobile, certain elements like table columns could be further optimised for smaller screens.
  - The administrators complained of the lack of CSS styling of the flask-admin page, which is admittedly on our company due to lack of foresight.
  - Multiple users recommended refining the colour palette with a more monotone or neutral theme to reduce eye strain during night-time use.

<u>Boundary Testing</u>
Boundary testing comparing actual output with expected output.

Function	Default value	Expected output	Actual output	Reason for inclusion
register_user()	Empy Form	User is added to database, responds with success message and outputs User object	User(id = .., email =...., etc.	Part of the key functional requirements
hw_solutions()	None	A row that providse the title and link of a hw solution pdf	A row that providse the title and link of a hw solution pdf	Integral to the key functional requirements