QUALIFICATION SPECIFICATION

gateway qualifications

Web Application Development Level 5 Diploma

Access to HE
Apprenticeships
Digital
Employability & Enterprise
English & Maths
ESOL
Personal & Social Development
Professional Development
Vocational

This qualification specification covers the following qualification:

Qualification Number	Qualification Title
603/5574/8	Gateway Qualifications Level 5 Diploma in Web Application Development

Version and date	Change detail	Section/Page Reference
2.2 July 2024	1. Back End Development unit aim has had ‘or non-relational’ removed 2. LO1 has had ‘micro-‘ removed 3. AC1.3 has had reference to ‘Flask’ removed 4. AC2.2 has had ‘or non-relational removed’	Back End Development, Page 44-46
2.1 March 2023	1. ‘About the qualification’ updated 2. ‘Key information’ table formatted 3. Support materials and resources section updated 4. Section updated 5. Section updated	1.1, Page 7 2.1, Page 9 4.4, Page 16 5.1, Page 19 5.2, Page 19
2.0 August 2022	1. 2nd bullet of 3.2 Qualification size amended. 2. Unit table updated 3. Resubmission of assessments section added 4. Support materials and resources section updated	3.2, Page 11 3.3 & 3.4, Pages 12 and 13 3.5, Page 14 4.4, Page 16

	5. Awarding section updated 6. Units updated within appendices 7. Gateway Qualifications address removed	7.2, Page 24 10, Page 27-65 8, Page 25 and back cover
1.4 Nov 2020	Tutor Requirements updated.	5.3, Pg 18
1.3 July 2020	GLH corrected.	3.2 Pg 11
1.2 July 2020	Training requirement added.	5.2 Pg 19
1.1 April 2020	Example 1 grade calculations corrected.	Pg 14
1.0 Month Year (March 2020)	n/a	n/a

About this qualification specification

This qualification specification is intended for tutors, internal quality assurers, centre quality managers and other staff within Gateway Qualifications recognised centres and/or prospective centres.

It sets out what is required of the learner in order to achieve the qualification. It also contains information specific to managing and delivering the qualification including specific quality assurance requirements.

The guide should be read in conjunction with the Gateway Qualifications Centre Handbook and other publications available on the website which contain more detailed guidance on assessment and quality assurance practice.

In order to offer this qualification, you must be a Gateway Qualifications recognised centre and be approved to offer the qualification.

If your centre is not yet recognised, please contact our Development Team to discuss becoming a Gateway Qualifications Recognised Centre:

Telephone: 01206 911211
Email: enquiries@gatewayqualifications.org.uk
Website: www.gatewayqualifications.org.uk/recognition

Contents

1. Qualification Information

1.1 About the qualification

The qualification has been approved by the Office of Qualifications and Examinations Regulation (Ofqual) that regulates qualifications, examinations and assessments in England and Qualifications Wales, the regulator of non-degree qualifications and the qualifications system in Wales.

It has been developed in collaboration with Code Institute. To support recognised centres, Code Institute has an optional comprehensive learner management system and learning content aligned to the qualification. Code Institute is recognised within the industry as a key provider in developing the necessary skills needed for careers in technology and software development and has consulted with employers including Accenture and Red Hat to ensure that the content remains relevant.

The qualification aims to develop:

- knowledge and skills in the design, development and testing of code solutions to information processing problems
- competence in the application of frameworks and code libraries to optimise the production and testing of code solutions
- knowledge and understanding of systems for storing data and acquire skills in accessing and integrating data from various sources
- an understanding of the characteristics of user interfaces and implement solutions that meet user requirements
- competence in the deployment of web-based information systems that are secure
- competence in working in a team managing a project to deliver a coding solution.

1.2 Purpose

The qualification purpose is to upskill those already working in a website development role but also to offer learners with no previous experience in programming a pathway to employment in this occupational area.

1.3 Funding

For information regarding potential sources of funding in England please visit the Education and Skills Funding Agency:

https://www.gov.uk/government/organisations/education-and-skills-funding-agency

https://www.gov.uk/government/publications/advanced-learner-loans-qualifications-catalogue

https://hub.fasst.org.uk/Pages/default.aspx

For information regarding potential sources of funding in Wales please visit Qualification Wales

https://www.qualificationswales.org

1.4 Geographical coverage

This qualification is approved by Ofqual to be offered in England and by Qualification Wales to be delivered in Wales.

If a centre based outside England or Wales would like to offer this qualification, they should make an enquiry to Gateway Qualifications.

1.5 Progression opportunities

On completion of the qualification learners are competent to work in the following roles:

- Junior/Associate Full Stack Developer
- Junior/Associate Front end Developer
- Junior/Associate Back end Developer
- Junior/Associate Software Developer
- Junior/Associate QA Engineer
- Support Engineer

Areas of potential future specialisation/study include:

- Front end Frameworks.
- Systems Administration
- Data Analytics
- QA Engineering
- User Experience
- DevOps Engineering

1.6 Equality, diversity and inclusion

It is Gateway Qualifications' aim that there shall be equal opportunities within this organisation and in all the services it provides and within its recognised centres and via the services they provide and so meet the organisation's legal responsibilities to prevent discrimination.

It is the organisation's intention that there should be no discrimination on the grounds of a protected characteristic including age, disability, gender assignment, marriage and civil partnership, pregnancy and maternity, race, religion and belief, sex, sexual orientation. It is acknowledged that this is not an exhaustive list.

2. Learner Entry Requirements

2.1 Key information

Qualification Titles	
Age	19+
Prior qualifications or units	There are no requirements for learners to have achieved prior qualifications prior to undertaking this qualification.
Prior skills/knowledge/ understanding	There is no requirement for learners to have prior skills, knowledge or understanding. However, centres should review the prior qualifications and experience of each learner and consider whether they provide the necessary foundations to undertake this programme of study at level 5.
Restrictions	There are no restrictions to entry.
Initial Assessment	Learners must undergo an initial assessment to ensure that they have the necessary foundations to undertake a programme of study at level 5. See section 4.5 Support materials and resources.
Additional requirements/guidance	Learners must have an appropriate level of English to enable them to access the relevant resources and complete the assessments for each unit.

2.2 Access to qualifications for learners with disabilities or specific needs

Gateway Qualifications and recognised centres have a responsibility to ensure that the process of assessment is robust and fair and allows the learner to show what they know and can do without compromising the assessment criteria.
Gateway Qualifications has a duty to permit a reasonable adjustment where an assessment arrangement would put a disabled person at a substantial disadvantage in comparison to someone who is not disabled. Please refer to Section 4.6 Access Arrangement, Reasonable Adjustments and Special Considerations for further details.

2.3 Recruiting learners with integrity

Centres must recruit learners with integrity. They must ensure that learners have the correct information and advice on their selected qualification and that the qualification will meet their needs.

Centres must assess each potential learner and make justifiable and professional judgements about their potential to successfully complete the assessment and achieve the qualification. Such an assessment must identify, where appropriate, the support that will be made available to the learner to facilitate access to the qualification.

3 Qualification Details

3.1 Achievement methodology

The qualification will be awarded to learners who successfully achieve an approved combination of units through a portfolio of evidence that has been successfully verified and monitored through Gateway Qualifications' Quality Assurance process. Achievement is therefore determined by successful completion of unit assessment with no further requirement for additional/summative assessment.

This qualification is graded at unit and qualification level.

3.2 Qualification size

Qualification Title	Total Qualification Time	Guided Learning	Credit Value
Gateway Qualifications Level 5 Diploma in Web Applications Development	750	597	75

Total Qualification Time is the number of notional hours which represents an estimate of the total amount of time that could be reasonably expected to be required for a learner to achieve and demonstrate the achievement of the level of attainment necessary for the award of the qualification.

Total Qualification Time is comprised of the following two elements:

- the number of hours which an awarding organisation has assigned to a qualification for Guided Learning, and
- an estimate of the number of hours a learner will reasonably be likely to spend in preparation, study or any other form of participation in education or training, including assessment, unlike Guided Learning, not under the Immediate Guidance or Supervision of a lecturer, supervisor, tutor or other appropriate provider of education or training.

3.3 Qualification structure

The qualification requirements are provided below.

The knowledge, skills and understanding that will be assessed as part of the qualification are set out within unit specifications. Unit contents, including the learning outcomes and associated assessment criteria, are contained within this specification and published on the Gateway Qualifications website.

For information on Recognition of Prior Learning/Exempt and Equivalent units please see section **3.5 Recognition of Prior Learning (RPL)**

Gateway Qualifications Level 5 Diploma in Web Application Development

Learners must complete units 1, 2, 3 and 4.

Unit	Unit Number	Unit Title	Level	Guided Learning	Credit Value
1	Y/650/3525	User Centric Front End Development	5	102	12
2	A/650/3526	Interactive Front End Development	5	152	16
3	D/650/3527	Back End Development	5	153	22
4	F/650/3528	Full Stack Frameworks with Django	6	190	25

3.4 Grading

The qualification is awarded as Pass/Merit/Distinction.

To achieve a merit or distinction grade, the learners must demonstrate that they have achieved all the criteria set for these grades. Where work for the pass standard is marginal, assessors can take account of any extension work completed by the learners.

To achieve a Pass	• learners must evidence all Pass criteria from the assessment and grading grid
To achieve a Merit	• learners must evidence all Pass and Merit criteria from the assessment and grading grid • partial achievement of the Merit criteria **cannot** attract the Merit grade.
To achieve a Distinction	• learners must evidence all Pass and Merit criteria and meet the requirements for a Distinction. • the requirements for a Distinction are qualitative extensions of the Merit criteria

The qualification grade will be automatically calculated for learners when the learner unit grades are submitted by a centre. The overall grade is calculated based on the rules of combination for the qualification, in the following way:

1. The grade is converted to a number of points per credit (see table below).

2. The number of points are totalled and the overall grade applied according to the 'qualification grade' table.

The table below shows the **number of points scored per credit** at the unit level and grade:

	Points per credit		
	Pass	Merit	Distinction
Levels 5 and 6	4	6	8

Learners who achieve the correct number of points within the ranges show in the 'qualification grade' table below will achieve the qualification merit or distinction grade:

	Pass	Merit	Distinction
Points range	300-381	382-487	488-600

Example 1

Achievement of pass qualification grade:

Units	**Credit**	**Grade**	**Grade Points**	**Total Unit Points** (credit x grade)
User Centric Front End Development	12	Merit	6	72
Interactive Front End Development	16	Pass	4	64
Back End Development	22	Merit	6	132
Full Stack Frameworks with Django	25	Pass	4	100
Totals	**75**			**368**

Overall grade: pass

Example 2

Achievement of merit qualification grade:

Units	**Credit**	**Grade**	**Grade Points**	**Total Unit Points** (credit x grade)
User Centric Front End Development	12	Pass	4	48
Interactive Front End Development	16	Pass	4	64
Back End Development	22	Merit	6	132
Full Stack Frameworks with Django	25	Merit	6	150
Totals	**75**			**394**

Overall grade: merit

3.5 Resubmission of assessments

Resubmission
The process of resubmitting evidence against the assessment brief where one or more of the learning outcomes and corresponding assessment criteria have not been met.

Following formal assessment, where learners have not met the required standard for one or more learning outcomes/assessment criteria, they may be given the opportunity to resubmit. A learner may be approved for a resubmission when:

- they have met the centre requirements for internally assessed work
- the tutor/assessor judges that the learner will be able to meet the learning outcomes and their corresponding assessment criteria independently

In addition to this, if possible, we would recommend that the centre has in a place a process where the internal quality assurer is consulted and/or approves the resubmission.

Centres should have a resubmission procedure available to learners that explains:

- the process learners should follow
- what learners can expect
- timescales
- how the learner can appeal

This must be included and explained in the learner handbook and the induction process.
If a resubmission is authorised, the centre must:

- record resubmissions on the learner assessment record
- state a deadline for resubmission that is proportionate to the size of the qualification
    - we recommend 15 working days of the learner receiving the results of the assessment. This may be adjusted to reflect the size of the units concerned
- inform the learner which learning outcomes and their corresponding assessment criteria have not been met
- provide information and guidance available to the learner that they could have used.

A learner should not be allowed to resubmit more than twice.

Where there are specific requirements relating to resubmission approaches, this will be in the qualification specification.

Resubmission of graded assessments

When marking graded assessments, the centre must not:

- penalise learners by capping the grade of the assessment
- use resubmissions to improve grades.

3.6 Recognition of prior learning

Recognition of prior learning is a process that considers if a learner can meet the specified assessment requirements through knowledge, understanding or skills that they already possess and that can contribute towards the attainment of a qualification which they are undertaking.

The process of Recognition for Prior Learning is not applicable to this qualification.

3.7 Links to other qualifications

There are no direct links to other qualifications.

4 Assessment

4.1 Assessment overview

The method of assessment for the qualification is through a portfolio of evidence. The portfolio of evidence usually consists of 4 projects.

4.2 Assessment format

Centres will need to provide learners access to an integrated development environment and online hosting environment for assessment purposes.

4.3 Assessment language

The qualifications are assessed in English only.

4.4 Support materials and resources

In addition to this qualification specification, the following resources are available on the Gateway Qualifications website:

- Centre Handbook

External Resources

Centres are strongly recommended to use the learner management system and content provided by Code Institute to support teaching, learning and qualification delivery. Assessments will be completed outside the learner management system using a suitable integrated development environment and online hosting environment.

Additional optional complimentary resources provided by Code Institute to centres include;

- diagnostic tool
- coding challenge
- train the trainer
- marketing support for centres
- integrated development environment and host environment that are a practice hub and assessment environment for learners

4.5 Access Arrangements, Reasonable Adjustments and Special Considerations

Gateway Qualifications and recognised centres have a responsibility to ensure that the process of assessment is robust and fair and allows the learner to show what they know and can do without compromising the assessment criteria. Gateway Qualifications understands

its requirement as an awarding organisation to make reasonable adjustments where a learner, who is disabled within the meaning of the Equality Act 2010, would be at a substantial disadvantage in comparison to someone who is not disabled.

Gateway Qualifications has identified reasonable adjustments permissible as detailed below. A reasonable adjustment is unique to an individual and therefore may not be included in the list of available access arrangements.

Centres do not need to apply to Gateway Qualifications for approval of reasonable adjustments unless adaptation of externally set assessments is required.

Learners can have access to all forms of equipment, software and practical assistance, such as a reader or a scribe that reflect their normal way of working within the centre. However, such adjustments must not affect the reliability or validity of assessment outcomes or give the candidate an assessment advantage over other candidates undertaking the same or similar assessments.

The following adaptations are examples of what may be considered for the purposes of facilitating access, as long as they do not impact on any competence standards being tested:

- adapting assessment materials;
- adaptation of the physical environment for access purposes;
- adaptation to equipment;
- assessment material in an enlarged format or Braille;
- assessment material on coloured paper or in audio format;
- British Sign Language (BSL);
- changing or adapting the assessment method;
- changing usual assessment arrangements;
- extra time, e.g. assignment extensions;
- language modified assessment material;
- practical assistant;
- prompter;
- providing assistance during assessment;
- reader;
- scribe;
- transcript;
- use of assistive software;
- using assistive technology;
- use of coloured overlays, low vision aids;
- use of a different assessment location;
- use of ICT/responses using electronic devices.

It is important to note that not all the adjustments (as above) will be reasonable, permissible or practical in particular situations. The learner may not need, nor be allowed the same adjustment for all assessments.

Learners should be fully involved in any decisions about adjustments/adaptations. This will ensure that individual needs can be met, whilst still bearing in mind the specified assessment criteria for a particular qualification.

All reasonable adjustments made by the centre must be recorded on the Gateway Qualifications' Reasonable Adjustments Form and should be made available to Gateway Qualifications upon request. Guidance on the process for applying for formal adjustments can be found on the Forms and Guidance page of Gateway Qualifications' website.
All adjustments to assessment/s must be authorised by the centre's named quality assurance nominee or a member of staff with delegated authority where a centre is permitted to make reasonable adjustments, i.e. for internally marked assessments.

Centres should keep records of adjustments they have permitted and those they have requested from Gateway Qualifications. These records should normally be kept for 3 years following the assessment to which they apply.

It is recommended that centres nominate members of staff to take responsibility for demonstrating the implementation and recording of adjustments to assessments for monitoring by Gateway Qualifications or the regulatory authorities.

Special Considerations

Requests for special consideration should be submitted as soon as possible after the assessment and no later than 5 working days after the assessment. Please refer to the Reasonable Adjustments and Special Consideration Policy for circumstances where requests for special consideration may be accepted after the results of assessment have been released.

5 Centre Recognition and Qualification Approval

5.1 Centre Recognition

Both centre recognition and qualification approval must be gained before centres are permitted to deliver this qualification.

Guidance on the centre recognition and qualification approval processes is available on the website: https://www.gatewayqualifications.org.uk/advice-guidance/help-admin-tasks/centre-recognition/

5.2 Centre requirements

Centres must ensure that they have the appropriate resources in place to deliver this qualification.

Centres must also provide learners with access to both an integrated development environment and host environment for the final assessments.

5.3 Qualification-specific tutor/assessor requirements

Tutor/Assessors must;

- Have hands-on industry experience in modern web technologies and have practical experience with imperative development languages, ideally Python, however any of the following will also suffice: C; C++; Java; Ruby; Perl;
- Be up to date with Front End technologies and trends – including HTML5, JavaScript, Responsive Design, Application Programming Interfaces
- Be up to date with Back End technologies and trends – ideally would include Python, Django, Database Design and test-drive development. Experience with PostgreSQL or MongoDB or an equivalent SQL and no-SQL Platform is required.
- Minimum 3 years web development experience

Internal Quality Assurers in addition to being Tutor/Assessors will have knowledge and experience of carrying out internal quality assurance/verification and will hold a recognised internal quality assurance/verification or be working towards one, examples as follows;

- D34 qualification
- V1 qualification
- Internal Verify Award
- Internal Verification of Credit Based Learning: Continuing Professional Development for Practitioners Award
- Level 4 Award in the Internal Quality Assurance of Assessment Processes and Practice
- Level 4 Certificate in Leading the Internal Quality Assurance of Assessment Processes and Practice
- L4 TAQA

6 Quality Assurance

Centres should refer to the online Centre Handbook for further guidance.

The quality assurance process for these qualifications is through risk-based external quality assurance monitoring through reviews of centres' internal quality assurance systems against key quality standards and sampling of assessment decisions and internal quality assurance activity to ensure that qualification standards are maintained.

Centre monitoring is undertaken by an external quality assurer (EQA) allocated to the centre. The EQA plays a critical role in the Gateway Qualifications approach to centre assessment standards scrutiny as they are responsible for:

- carrying out an annual compliance visit
- validating the centre's procedures for delivery of qualifications and assessment
- completing reports for each visit with clear action points where needed
- risk rating centres on the above.

The EQA carries out an initial risk assessment at the centre recognition stage and then annually on an on-going basis using Gateway Qualifications' risk assessment criteria, and gives a high/medium/low risk rating in each of the following categories:

- centre resourcing and arrangements: this includes consideration of centre staffing, induction and training, policies and compliance with our centre agreement
- internal assessment and delivery: including reference to staff knowledge and skills, understanding of requirements, and appropriateness of delivery arrangements; also, delivery of external assessments including invigilation, conduct of assessments and confidentiality (where appropriate)
- internal quality assurance: covering IQA procedures, whether staff are appropriately trained, and standardisation arrangements are in place
- learner experience: that embraces appropriateness of initial assessment and learners being on the correct programme, learner induction and course support.

EQAs arrange quality monitoring visits to all recognised centres. These visits:

- monitor the centre's compliance with the centre recognition terms and conditions by reviewing programme documentation and meeting managers and centre staff
- identify any staff development needs
- ensure that all procedures are being complied with, through an audit trail, and make sure that the award of certificates of completion to learners is secure.

EQAs contact the centre in advance of a visit, however Gateway Qualifications reserves the right to undertake unannounced visits including during assessment times.

EQAs will request information from the centre in advance of a planned visit to help inform the evidence to be reviewed during the visit. Centres are obliged to comply with any requests for access to premises, people and records for the purposes of the monitoring visit. If a centre fails to provide access, then Gateway Qualifications will take appropriate action.

Once a visit date has been agreed, the centre should ensure that the appropriate members of staff attend the meeting, all requested documentation is provided and access to qualification, learner and staff records is available.

If a centre cancels a pre-arranged monitoring visit at short notice the EQA must be satisfied that there was a legitimate reason for the cancellation. If this cannot be established, Gateway Qualifications reserves the right to withhold certification claims until a monitoring visit is completed.

Following the visit, the EQA completes a monitoring report which will be sent to the centre for reference afterwards.

As part of ongoing monitoring EQAs will need access to the learning environment to conduct sampling activities which may be done remotely.

The frequency of the quality monitoring will be determined by the volume of learner registrations and the actions arising from previous monitoring activity.

Centres found in breach of these procedures may be subject to sanctions by Gateway Qualifications. Please refer to the Gateway Qualifications Sanctions Policy.

6.1 Internal Quality Assurance

As the assessments are tutor marked the centre must operate an internal quality assurance process. This ensures that qualification standards are being applied consistently within a centre through training, standardisation, sampling of marking and feedback. A centre's internal quality assurance process is led by the Internal Quality Assurer (IQA) who is responsible for ensuring that all tutors are marking assessments in line with the standards set by Gateway Qualifications.

Internal Standardisation
Internal standardisation is a collaborative process by which tutors within a centre consider work that they have marked and, using pre-determined criteria, reach a common agreement on standards as being typical of work at a particular level and grade by comparing samples and providing peer evaluation.

Standardisation will be facilitated by the Centre's IQA and should include all those involved in marking assessments. Centre standardisation events should be held at regular intervals and to a schedule which reflects delivery patterns and supports the marking of live assessments. Centres will be required to keep records of each internal standardisation event including the date, attendees and notes on any outcomes and actions. Centres will be required to store these reports securely for three years and Gateway Qualifications may ask to see these records as part of the centre quality assurance and monitoring activities.

6.2 Quality assuring centre marking

Once the internal quality assurance process is complete, an EQA will be allocated to a centre to sample the centre marking.

The sample selected is based on the number of learners and the centre's risk rating, derived from centre monitoring.

Evidence of the inconsistent marking and actions taken informs the centre's risk rating and this information will be taken into account with the sampling of future assessments, for example, leading to an increase in sampling size.

6.3 Malpractice

Malpractice is any deliberate activity, neglect, default or other practice that compromises the integrity of the internal and external assessment process, and/or the validity of certificates. It covers any deliberate actions, neglect, default or other practice that compromises, or could compromise:

- the assessment process
- the integrity of a regulated qualification
- the validity of a result or certificate
- the reputation and credibility of Gateway Qualifications
- the qualification to the public at large.

Centre staff should be familiar with the contents of Gateway Qualifications Malpractice and Maladministration Policy, https://www.gatewayqualifications.org.uk/advice-guidance/delivering-our-qualifications/centre-handbook/quality-compliance/

6.4 Additional quality assurance requirements

Direct Claims is not permitted for this qualification therefore all certification claims must be validated by the EQA.

7 Learner Registration and Results

7.1 Registration

Centres will register learners via the online registration portal. Learner registration guidance is available on our website, https://www.gatewayqualifications.org.uk/advice-guidance/help-admin-tasks/registering-learners/.

7.2 Awarding

The qualifications will be awarded as Pass, Merit, Distinction. Learners must meet the learning outcomes and corresponding assessment criteria for all four units and will be awarded a Pass as a minimum. To be awarded higher grades please refer to unit details.

7.3 Issuing results

Results for learners who do not reach the minimum standard for a Pass will be recorded as Fail.

7.4 Appeals

Centres must have internal appeal arrangements which learners can access if they wish to appeal against a decision taken by Centres, which will include a named contact at the Centre. These arrangements have to be transparent and accessible in order that appeals from learners can be received, considered and resolved fairly.
Please refer to the Gateway Qualifications' Appeals policy:
https://www.gatewayqualifications.org.uk/wp-content/uploads/2017/09/Appeals-Policy.pdf

7.5 Enquiries

Enquiries about assessment decisions should be made once the centre has followed its internal enquiries and appeal procedures.

Contact details are available on our website:
https://www.gatewayqualifications.org.uk/contact-us/

8 What to do next

For existing centres please contact your named Development Manager or Development Officer.

For organisations, not yet registered as Gateway Qualifications centre please contact:

Tel: 01206 911211
Email: enquiries@gatewayqualifications.org.uk

9 Gateway Qualifications

Gateway Qualifications, a not for profit registered charity, is an Awarding Organisation based in Colchester.

We work with learning providers and industry experts to design and develop qualifications that benefit the learner and the employer.

We support flexible, responsive and quality assured learning opportunities whether it is in the classroom, at work, in the community or through distance learning.

We are recognised by Ofqual, to design, develop and submit qualifications to the Regulated Qualifications Framework (RQF) and Qualification Wales to offer regulated qualifications in Wales.

10 Appendices

10.1 Appendix 1 – Unit Details

Unit 1: User Centric Front End Development

Level:	Level 5
Credit Value:	12
GLH:	102
Unit Number:	Y/650/3525
Unit Aim:	This unit aims to provide learners with the knowledge and skills needed to build a Front end web application. It includes understanding the principles of responsive design, documentation and effective layout of content.

This unit has 5 learning outcomes.

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
1 Design a Front end web application based on the principles of user experience design, accessibility and responsivity.	1.1 Design a website that incorporates a main navigation menu and a structured layout. 1.2 Design a website that meets accessibility guidelines (e.g. contrast between background and foreground colours, non-text elements have planned alternative text equivalents to cater for the visually impaired). 1.3 Design the organisation of information on the page following	M(i) Design a website with a flow of information layout, and interaction feedback which are clear and unambiguous.	Please refer to the performance characteristics for distinction.**

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	the principles of user experience design (headers are used to convey structure, information is easy to find due to being presented and categorised in terms of priority). 1.4 Ensure that foreground information is never distracted by backgrounds. 1.5 Include graphics that are consistent in style and colour. 1.6 Design the site to allow the user to initiate and control actions such as pop-ups and playing of audio/video.		
2 Develop and implement a static Front end web application using HTML and CSS.	2.1 Create a website of at least 3 pages, or (if using a single scrolling page) at least 3 separate page areas, to match the design and to meet its stated purpose. 2.2 Write custom CSS code that passes through the official (Jigsaw) validator with no issues. 2.3 Write custom HTML code that passes through the official W3C validator with no issues. 2.4 Incorporate images that are of sufficient resolution to not appear pixelated or stretched.	M(ii) Implement a website whose purpose is immediately evident to a new user without having to look at supporting documentation. M(iii) Implement a website that provides a solution to the user story demands and expectations.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	2.5 Code all external links to open in a separate tab when clicked. 2.6 Use CSS media queries or CSS Grid/Bootstrap across the application to ensure the layout changes appropriately and maintains the page's structural integrity across device screen sizes. 2.7 Use Semantic markup to structure HTML code. 2.8 Present the finished website with clearly understandable site-specific content, rather than Lorem Ipsum placeholder text. 2.9 Implement clear navigation to allow users to find resources on the site intuitively.		
3 Maximise future maintainability through documentation, code structure and organisation.	3.1 Write a README.md file for the web application that explains its purpose, the value that it provides to its users, and the deployment procedure. 3.2 Insert screenshots of the finished project that align to relevant user stories. 3.3 Attribute all code from external sources to its original source via comments above the code and (for larger dependencies) in the README.		

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	3.4 Clearly separate and identify code written for the website and code from external sources (e.g. libraries or tutorials). 3.5 Organise HTML and CSS code into well-defined and commented sections. 3.6 Place CSS code in external files, linked to the HTML page in the HEAD element. 3.7 Write code that meets at least minimum standards for readability (consistent indentation, blank lines only appear individually or, at most, in pairs). 3.8 Name files consistently and descriptively, without spaces or capitalisation, to allow for cross-platform compatibility. 3.9 Group files in directories by file type (e.g. an assets directory will contain all static files and may be organized into sub-directories such as CSS, images, etc.)		
4 Use version control software to maintain, upload and share code with other developers.	4.1 Use a cloud-based, git-based, version control system (e.g. Git & GitHub) throughout the development and implementation process.	M(iv) Commit often, for each individual feature/fix, ensuring that commits are small and well-defined, with clear, descriptive messages.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	4.2 Document the development process through descriptive commit messages. 4.3Use consistent and effective markdown formatting to produce a README file that is well-structured, easy to follow, and has few grammatical errors.		
5 Test and deploy a Front end web application to a Cloud platform.	5.1 Design and implement manual testing procedures to assess functionality, usability and responsiveness. 5.2 Document the testing in the README or in a separate file. 5.3 Deploy a final version of the code to a cloud-based hosting platform (e.g. GitHub Pages) and test to ensure it matches the development version. 5.4 Remove commented-out code before pushing final files to version control and deploying. 5.5 Ensure that there are no broken internal links.	M(v Present a clear rationale for the development of the project, in the README, demonstrating that it has a clear, well-defined purpose addressing the needs of, and user stories for a particular target audience (or multiple related audiences). M(vi) Document testing fully to include evaluation of bugs found and their fixes and explanation of any bugs that are left unfixed. M(vii) Fully document the development life cycle procedures in the README file.	

*Additional guidance for merit

To achieve merit learners, need to meet the assessment criteria outlined above for a pass and a merit.

The following additional guidance describes characteristics of performance at MERIT.

The learner has a clear rationale for the development of this project and has produced a fully functioning, well-documented, website for a real-life audience.

The finished project has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences). Its purpose would be immediately evident to a new user without having to look at supporting documentation. The design of the web site follows the principles of UX design and accessibility guidelines, and the site is fully responsive.

Code is well-organised and easy to follow, and the application has been fully tested, following a planned, manual testing procedure, with no obvious errors left in the code.

The development process is clearly evident through commit messages. The project's documentation provides a clear rationale for the development of this project and covers all stages of the development life cycle.

**Characteristics of Performance at Distinction

To achieve a distinction, a learner will have achieved all pass and merit criteria, as described above, and will demonstrate characteristics of high level performance as described below:

Characteristics of performance at DISTINCTION:

The learner has documented a clear, **justified**, rationale for a real-world application and a comprehensive explanation of how it will be developed. The development of the project has resulted in a fully-functioning, interactive, web application, developed using at least one more advanced technique (e.g. CSS media queries)

The finished project is judged to be publishable in its current form with a clearly evidenced professional grade user interface and interaction adhering to current practice. There are no obvious errors in the code. Where there is a clear breach of accepted design/UX principles, or of accepted good practice in code organisation, these are fully justified, appropriate and acceptable to the target user. It clearly matches the design and demonstrates the characteristics of **craftsmanship** in the code.

Code is clearly signposted and there is clear, documented evidence of testing at all stages of development. There is full evidence of end testing, and there is an evaluation of bugs that may remain in the code.

The development and testing process is clearly evident, **and justified**, through commit messages. The project's documentation provides a **clear and detailed** rationale for the development of this project, covering **all stages** of the development life cycle.

Amplification (craftsmanship) This amplification is only applicable to performance at distinction.

Design

The design of the web application demonstrates the main principles of good UX design:

- Information Hierarchy
  - headers are used to convey structure - each section has a header that is easy to see and clear to understand
  - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all resources on the site are easy to find, allowing users to navigate the layout of the site intuitively
  - information is presented and categorised in terms of its priority
- User Control
  - all interaction with the site would be likely to produce a positive emotional response within the user. This is down to the flow of information layout, the clear and unambiguous navigation structures and all interaction feedback
  - when displaying media files, the site avoids aggressive automatic pop-ups and autoplay of audio; instead letting the user initiate and control such actions
- Consistency
  - evident across all pages/sections and covers interactivity as well as design
- Confirmation
  - user actions are confirmed where appropriate, feedback is given at all times
- Accessibility
  - there is clear conformity to accessibility guidelines across all pages/sections and in all interactivity

Any design decisions that contravene accepted user interaction, user experience design principles are **identified and described** (comments in code and/or a section in the README)

Development and Implementation

Code demonstrates accepted standards for readability (including characteristics of ‘clean code’):

- Consistent and appropriate naming conventions within code and in file naming, e.g.
  - file names and class names, are descriptive and consistent
  - for cross-platform compatibility, file and directory names will not have spaces in them and will be lower-case only
  - all HTML attributes and CSS rules, are consistent in format, follow standards for the language and are appropriate and meaningful
- File structure e.g.
  - whenever relevant, files are grouped in directories by file type (e.g. an assets’ directory will contain all static files and code may be organised into sub-directories such as CSS, etc)
  - there is a clear separation between custom code and any external files (for example, library files are all inside a directory named 'libraries')

    - files are named consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Readability
    - code is indented in a consistent manner to ease readability and there are no unnecessary repeated blank lines (and never more than 2)
    - function/class/variable names clearly indicate their purpose
    - CSS code is split into well-defined and commented sections
    - Semantic mark-up is used to structure HTML code
    - HTML and CSS are kept in separate, linked files
    - CSS files are linked in the HTML file's head element
- Defensive design
    - any potential user errors are identified and dealt with
- Comments
    - all custom code files include clear and relevant comments explaining the purpose of code segments
- Compliant code
    - HTML code passes through the official W3C validator with no issues
    - CSS code passes through the official (Jigsaw) validator with no issues
    - JavaScript code passes through a linter (e.g. jslint.com) with no major issues
- Robust code
    - no logic errors are found when running code
    - errors caused by user actions are handled
    - where used, API calls that fail to execute or return data will be handled gracefully, with the site users notified in an obvious way
    - inputs are always validated.

The full design is implemented providing **a good solution to the users' demands and expectations**.

Real world application

- Clearly understandable site-specific content is used rather than Lorem Ipsum placeholder text
- The final application is aligned to the user stories presented at the start of the project

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. All noticeable errors have been corrected or documented.

Version control systems are used effectively:

- all code is managed in git with well-described commit messages
- there is a separate, well-defined commit for each individual feature/fix
- there are no very large commits which make it harder to understand the development process and could lead the assessor to suspect plagiarism

The full application development process is documented:

- the purpose of the application is clearly described in the README
- the project's documentation describes the UX design work undertaken for this project and the reasoning behind it
    - wireframes, mockups, diagrams, etc., created as part of the design process are included in the project

- there is a clear separation between code written by the learner and code from external sources (e.g. libraries or tutorials). All code from external sources is attributed to its source via comments above the code and (for larger dependencies) in the README
- the testing procedure is well documented either in the README or a separate file
- the deployment procedure is fully documented in a section in the README file
- clear, well-described, explanatory commit messages describe the development process
- the README is well-structured and easy to follow
- the README file is written in markdown and uses markdown formatting consistently and effectively
- project documentation and the application's user interface have few errors in spelling and grammar.

Unit 2: Interactive Front End Development

Level:	Level 5
Credit Value:	16
GLH:	152
Unit Number:	A/650/3526
Unit Aim:	This unit aims to provide learners with the knowledge and skills needed to build a dynamic, interactive Front end web application. It includes understanding user input and control when interacting with a web application. It also includes understanding the principles of automated and manual testing for debugging.

This unit has 5 learning outcomes.

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
1 Design, develop and implement a dynamic Front end web application using HTML, CSS and JavaScript.	1.1 Design a web application that meets accessibility guidelines, follows the principles of UX design and presents a structured layout and navigation model, and meets its given purpose. 1.2 Design interactivity for a web application that lets the user initiate and control actions and gives feedback. 1.3 Write custom JavaScript, HTML and CSS code to create a responsive Front end web application consisting of one or more HTML pages with	M(i) Design a web application following the principles of UX design which meets accessibility guidelines, is easy to navigate and allows the user to find information and resources intuitively.	Please refer to the performance characteristics for distinction.**

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	significant interactive functionality. 1.4 Write JavaScript code to produce relevant responses to user actions. 1.5 Implement an interactive web application that incorporates images or graphics of usable resolution, legible, unobscured text, consistent styling, undistracted foregrounds.		
2 Implement Front end interactivity, using core JavaScript, JavaScript libraries and/or Application Programming Interfaces (APIs).	2.1 Write JavaScript code, that passes through a linter (e.g. JSLint) with no major issues and write validated HTML and CSS code. 2.2 Write JavaScript functions that correctly implement compound statements such as “if statements” and/or loops. 2.3 Write code that intelligently handles empty or invalid input data. 2.4 Implement appropriate working functionality for all project requirements. 2.5 Organise non-trivial JavaScript code in external file(s) linked at the bottom of the body element (or bottom of head element if needs to be loaded before the	M(ii) Write code such that users who direct to a non-existent page or resource are redirected back to the main page without having to use browser navigation buttons.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	body HTML) and CSS code in external files linked to HTML in the head element. 2.6 Write code that meets minimum standards for readability (comments, indentation, consistent and meaningful naming conventions). 2.7 Name files consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility. 2.8 Write code that does not generate internal errors on the page or in the console as a result of user actions. 2.9 Organise code and assets files in directories by file type.		
3 Test an interactive Front end web application through the development, implementation and deployment stages.	3.1 Explain the principles of automated and manual testing and when each might be deployed. 3.2 Design and implement testing procedures (automated or manual) to assess functionality, usability and responsiveness of the web application. 3.3 Insert screenshots of the finished project that align to relevant user stories.		

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	3.4 Apply test procedures during development and implementation stages and test to ensure the deployed version matches the development version. 3.5 Fully document the results of well-planned testing procedures (automated or manual) to assess the website's functionality, usability and responsiveness.		
4 Deploy an interactive Front end web application to a Cloud platform.	4.1 Deploy a final version of the interactive web application code to a cloud-based hosting platform (e.g. GitHub Pages). 4.2 Ensure that the deployed application is free of commented out code and has no broken internal links. 4.3 Use Git & GitHub for version control of an interactive web application up to deployment.	M(iii) Commit often, for each individual feature/fix, ensuring that commits are small, well-defined and have clear, descriptive messages.	
5 Demonstrate and document the development process through a version control system such as GitHub.	5.1 Document the full development cycle, with clear evidence given through commit messages, the README. 5.2 Write a README.md file for the interactive web application that explains its purpose and the value that it provides to its users. 5.3 Clearly separate and identify code written for the interactive	M(iv) Present a clear rationale for the development of the project, in the README, demonstrating that it has a clear, well-defined purpose addressing the needs of, and user stories for a particular target audience	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	web application and code from external sources (e.g. libraries or tutorials). Attribute any code from external sources to its source via comments above the code and (for larger dependencies) in the README. 5.4 Use consistent and effective markdown formatting that is well-structured, easy to follow, and has few grammatical errors, when writing a README file.	(or multiple related audiences). M(v) Document the UX design work undertaken for this project, including any wireframes, mock-ups, diagrams created as part of the design process, and the reasoning behind it. Include diagrams created as part of the design process and demonstrate that these have been followed through to implementation. M(vi) Document testing fully to include evaluation of bugs found and their fixes and explanation of any bugs that are left unfixed. M(vii) Fully document the deployment procedure in a section in the README file.	

*Additional guidance for merit

To achieve merit learners, need to meet the assessment criteria outlined above for a pass and a merit.

The following additional guidance describes characteristics of performance at MERIT.

The learner has a clear rationale for the development of this project and has produced a fully functioning, well-documented, interactive, web application for a real-life audience, with specific content rather than placeholders. There is a range of interactive features. Data validation, API handling (when applied) and user feedback are all evident in the code and the working application. Optionally, a range of external and internal APIs (e.g. TMDb, Google Maps, etc.) have been used to produce working features. There are no logical errors in the code and the application functions as expected.

The finished project has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences). Its purpose would be immediately evident to a new user without having to look at supporting documentation. The user is kept informed of progress and actions through feedback and, where large data sets are being loaded, progress indicators. The design of the web application follows the principles of UX design and accessibility guidelines and the site is fully responsive.

Code is well-organised and easy to follow and the application has been fully tested, following manual or automated testing procedures, with no obvious errors left in the code. If automated testing is implemented unit test files should be present in code commits.

The project's documentation provides a clear rationale for the development of this project and covers all stages of the development life cycle.

**Characteristics of Performance at Distinction

To achieve a distinction a learner will have achieved all pass and merit criteria, as described above, and will demonstrate characteristics of high-level performance as described below:

Characteristics of performance at distinction:

The learner has documented a clear, justified, rationale for a real-world application and a comprehensive explanation of how it will be developed. The development of the project has resulted in a fully-functioning, interactive, web application. When used, the learner shows a clear understanding of asynchronicity and the timing problems that can arise when accessing shared data values.

The finished project is judged to be publishable in its current form with a clearly evidenced professional grade user interface and interaction adhering to current practice. There are no logic errors in the code. Where there is a clear breach of accepted design/UX principles, or of accepted good practice in code organisation, these are fully justified, appropriate and acceptable to the target user. It clearly matches the design and demonstrates the characteristics of craftsmanship in the code. The resulting application is original and not a copy of any walkthrough projects encountered in the unit

Amplification (craftsmanship) this amplification is only applicable to performance at distinction.

Design

The design of the web application demonstrates the main principles of good UX design:

- Information Hierarchy
  - headers are used to convey structure - each section has a header that is easy to see and clear to understand
  - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all resources on the site are easy to find, allowing users to navigate the layout of the site intuitively
  - information is presented and categorised in terms of its priority
- User Control
  - all interaction with the site would be likely to produce a positive emotional response within the user. This is down to the flow of information layout, the clear and unambiguous navigation structures and all interaction feedback
  - when displaying media files, the site avoids aggressive automatic pop-ups and autoplay of audio; instead letting the user initiate and control such actions
  - users who direct to a non-existent page or resource are redirected back to the main page without having to use the browser navigation buttons
  - the user is shown progress indicators where relevant
  - errors resulting from user actions are reported to the user
- Consistency
  - evident across all pages/sections and covers interactivity as well as design
- Confirmation
  - user actions are confirmed where appropriate, feedback is given at all times
- Accessibility
  - there is clear conformity to accessibility guidelines across all pages/sections and in all interactivity

Any design decisions that contravene accepted user interaction, user experience design principles are **identified and described** (comments in code and/or a section in the README)

Development and Implementation

Code demonstrates accepted standards for readability (including characteristics of ‘clean code’):

- Consistent and appropriate naming conventions within code and in file naming, e.g.
  - file names, class name, function names and variable names are descriptive and consistent
  - for cross-platform compatibility, file and directory names will not have spaces in them and will be lower-case only
  - all HTML attributes, CSS rules, code variables and function names are consistent in format, follow standards for the language and are appropriate and meaningful

- File structure e.g.
    - whenever relevant, files are grouped in directories by file type (e.g. an assets directory will contain all static files and code may be organised into sub-directories such as CSS, JavaScript, etc)
    - there is a clear separation between custom code and any external files (for example, library files are all inside a directory named 'libraries')
    - files are named consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Readability
    - code is indented in a consistent manner to ease readability and there are no unnecessary repeated blank lines (and never more than 2)
    - function/class/variable names clearly indicate their purpose
    - all code is split into well-defined and commented sections
    - functions are well-defined and commented to indicate purpose
    - Semantic markup is used to structure HTML code
    - HTML, CSS and JavaScript are kept in separate, linked files
    - CSS files are linked to in the HTML file's head element
    - non-trivial JavaScript code files are linked at the bottom of the body element (or bottom of head element if needs loaded before the body HTML)
- Defensive design
    - all input data is validated (e.g. presence check, format check, range check)
    - internal errors are handled gracefully, and users are notified of the problem where appropriate.
- Comments
    - all custom code files include clear and relevant comments explaining the purpose of code segments
- Compliant code
    - HTML code passes through the official W3C validator with no issues
    - CSS code passes through the official (Jigsaw) validator with no issues
    - JavaScript code passes through a linter (e.g. jslint.com) with no major issues
- Robust code
    - no logic errors are found when running code
    - errors caused by user actions are handled
    - where used, API calls that fail to execute or return data will be handled gracefully, with the site users notified in an obvious way
    - inputs are always validated.

The full design is implemented providing **a good solution to the users' demands and expectations**.

Real world application

- Clearly understandable site-specific content is used rather than Lorem Ipsum placeholder text
- Navigating between pages via the back/forward buttons can never break the site
- All links to external pages open in a separate tab when clicked
- The final application is aligned to the user stories presented at the start of the project

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. All noticeable errors have been corrected or documented:

- navigating between pages via the back/forward buttons can never break the site, there are no broken links
- user actions do not cause internal errors on the page or in the console
- if automated testing is implemented unit tests are included in code commits

Version control systems are used effectively:

- all code is managed in git with well-described commit messages
- there is a separate, well-defined commit for each individual feature/fix
- there are no very large commits which make it harder to understand the development process and could lead the assessor to suspect plagiarism

The full application development process is documented:

- the purpose of the application is clearly described in the README
- the project's documentation describes the UX design work undertaken for this project and the reasoning behind it
    - wireframes, mock-ups, diagrams, etc., created as part of the design process are included in the project
- there is a clear separation between code written by the learner and code from external sources (e.g. libraries or tutorials). All code from external sources is attributed to its source via comments above the code and (for larger dependencies) in the README
- the testing procedure is well documented either in the README or a separate file
- the deployment procedure is fully documented in a section in the README file
- clear, well-described, explanatory commit messages describe the development process
- the README is well-structured and easy to follow
- the README file is written in markdown and uses markdown formatting consistently and effectively
- project documentation and the application's user interface have few errors in spelling and grammar

Unit 3: Back End Development

Level: Level 5

Credit Value: 22

GLH: 153

Unit Number: D/650/3527

Unit Aim: This unit aims to provide learners with the knowledge and skills needed to build a Back end web application. Topics covered include data storage and data management using relational databases.

This unit has 5 learning outcomes.

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
1 Design, develop and implement a Back end for a web application using Python and a framework.	1.1 Design a Front end for a data-driven web application that meets accessibility guidelines, follows the principles of UX design, meets its given purpose and provides a set of user interactions. 1.2 Implement custom HTML and CSS code to create a responsive full-stack application consisting of one or more HTML pages with relevant responses to user actions and a set of data manipulation functions. 1.3 Build a database-backed web application that allows users to store and manipulate data	M(i) Design a Front end for a Full Stack application following the principles of UX design which meets accessibility guidelines, is easy to navigate and allows the user to find information and resources intuitively. (Mii) Design a Full Stack application that lets the user initiate and control actions and gives immediate and full feedback on data processes. M(iii) Implement a Full Stack application whose	Please refer to the performance characteristics for distinction.**

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	records about a particular domain. 1.4 Design a database structure that is relevant to the domain, including relationships between records of different entities. 1.5 Design and implement test procedures (automated or manual) to assess functionality, usability, responsiveness and data management within the Full Stack web application. 1.6 Write Python code that is consistent in style and conforms to the PEP8 style guide (or another explicitly mentioned style guide, such as Google's) and validated HTML and CSS code. 1.7 Write Python logic to demonstrate proficiency in the language. 1.8 Include functions with compound statements such as if conditions and/or loops in Python code. 1.9 Write code that meets minimum standards for readability (comments, indentation, consistent and meaningful naming conventions). 1.10 Name files consistently and descriptively, without spaces or	purpose is immediately evident to a new user and which provides a good solution to the user's demands and expectations. M(iv) Create templates, writing code that demonstrates understanding of template syntax, logic and usage. M(v) Write robust code that is free of errors in all parts of the application. M(vi) Fully document the results of well-planned testing procedures (automated or manual) to assess the website's functionality, usability and responsiveness. Include evaluation of bugs found and their fixes and explanation of any bugs that are left unfixed.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	capitalisation to allow for cross-platform compatibility.		
2 Model and manage data.	2.1 Design a data model that fits the purpose of the project. 2.2 Develop the model into a usable relational database where data is stored in a consistent and well-organised manner.	M(vii) Describe the data schema fully in the README file. M(viii) Maintain database configuration in a single location where it can be changed easily. M(ix) Maintain a Procfile, requirements.txt file, settings file.	
3 Query and manipulate data.	3.1 Create functionality for users to create, locate, display, edit and delete records.	M(x) Implement working Create, Read, Update and Delete (CRUD) M(xi) Check that Create, Read, Update and Delete (CRUD) actions are immediately reflected in the user interface.	
4 Deploy a Full Stack web application to a Cloud platform.	4.1 Deploy a final version of the full-stack application code to a cloud-based hosting platform (e.g. Heroku) and test to ensure it matches the development version. 4.2 Ensure that final deployed code is free of commented out code and has no broken internal links. 4.3 Document the deployment process in a README file that also explains the application's	M(xii) Commit often for each individual feature/fix, ensuring that commits are small, well-defined and have clear descriptive messages. M(xiii) Fully document the deployment procedure in a section in a README file, written using consistent and effective markdown formatting that	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	purpose and the value that it provides to its users.	is well-structured, easy to follow, and has few grammatical errors.	
5 Identify and apply security features.	5.1 Use Git & GitHub for version control of a Full Stack web application up to deployment, using commit messages to document the development process. 5.2 Commit final code that is free of any passwords or secret keys, to the repository and to the hosting platform. 5.3 Use environment variables, or files that are in gitignore, to hide all secret keys. 5.4 Ensure that DEBUG mode is turned off in production versions.	M(xiv) Present a clear rationale for the development of the project, in the README, demonstrating that it has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences), explaining the data, and explaining the security features considered.	

*Additional guidance for merit

To achieve merit learners, need to meet the assessment criteria outlined above for a pass and a merit.

The following additional guidance describes characteristics of performance at MERIT.

The learner has a clear rationale for the development of this project and has produced a fully functioning, well-documented, Database backed, Full Stack application for a real-life audience, with a full set of CRUD(creation, reading, updating and deletion of data records) features. There is a range of features, including creation, location, deletion and updating of data records. Data validation, and user feedback are all evident in the code and the working application. Templates have been used correctly produce working features. There are no logic errors in the code and the application functions as expected.

The finished project has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences) and a particular data domain. Its purpose would be immediately evident to a new user without having to look at supporting documentation. The user is kept informed of progress and actions through feedback and, where large data sets are being loaded, progress indicators. The design of the web application follows the principles of UX design and accessibility guidelines and the site is fully responsive.

Data is fully modelled and matches the schema. The schema design is documented in the README. Data store configuration is kept in a single location and can be changed easily. Configuration and settings files are well-organised and there are different versions for different branches.

Code is well-organised and easy to follow and the application has been fully tested, following a manual testing procedure, with no obvious errors left in the code.

The development process is clearly evident through commit messages. The project's documentation provides a clear rationale for the development of this project and covers all stages of the development life cycle.

The application is robust and deals with external errors gracefully (user input, API calls, asynchronous processes).

**Characteristics of Performance at Distinction

To achieve a distinction, a learner will have achieved all pass and merit criteria, as described above, and will demonstrate characteristics of high level performance as described below:

Characteristics of performance at distinction:

The learner has documented a clear, justified, rationale for a real-world application and a comprehensive explanation of how it will be developed. The development of the project has resulted in a fully-functioning, interactive, Full Stack application, with well-designed data and a full set of CRUD data operations. The learner shows a clear understanding of data modelling techniques and of the relationship between the Back end and Front end.

The finished project is judged to be publishable in its current form with a professional grade user interface and functionality, and interaction adhering to current practice. There are no logic errors in the code. Where there is a clear breach of accepted design/UX principles, or

of accepted good practice in code organisation, these are fully justified, appropriate and acceptable to the target user. It clearly matches the design and demonstrates the characteristics of craftsmanship in the code. The database schema is representative of complex user stories and there is a fully documented and full set of data operations which are fit for purpose in relation to the domain. The resulting application is original and not a copy of any walkthrough projects encountered in the unit

Amplification (craftsmanship) this amplification is only applicable to performance at distinction.

Design

The design of the web application demonstrates the main principles of good UX design:

- Information Hierarchy
  - semantic markup is used to convey structure - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all resources on the site are easy to find, allowing users to navigate the layout of the site intuitively
  - information is presented and categorised in terms of its priority
- User Control
  - all interaction with the site would be likely to produce a positive emotional response within the user. This is down to the flow of information layout, use of colour, clear and unambiguous navigation structures and all interaction feedback
  - when displaying media files, the site avoids aggressive automatic pop-ups and autoplay of audio; instead letting the user initiate and control such actions
  - users who direct to a non-existent page or resource are redirected back to the main page without having to use the browser navigation buttons
  - users are never asked for information that the application already has (e.g. a contact form does not ask a logged in user for an email address).
  - the user is shown progress indicators and feedback on transactions.
  - errors resulting from user or data actions are reported to the user
- Consistency
  - evident across all pages/sections and covers interactivity as well as design
  - consistency across all data operations, including in the reporting
- Confirmation
  - user and data actions are confirmed where appropriate, feedback is given at all times
- Accessibility
  - there is clear conformity to accessibility guidelines across all pages/sections and in all interactivity

Any design decisions that contravene accepted user interaction, user experience design principles are identified and described (comments in code and/or a section in the README)

Development and Implementation

Code demonstrates characteristics of 'clean code'**:**

- Consistent and appropriate naming conventions within code and in file naming, e.g.
  - file names, class names, function names and variable names are descriptive and consistent
  - for cross-platform compatibility, file and directory names will not have spaces in them and will be lower-case only
  - all HTML attributes, CSS rules, code variables and function names are consistent in format, follow standards for the language and are appropriate and meaningful
  - app urls are consistent
- File structure
  - whenever relevant, files are grouped in directories by file type (e.g. an assets directory will contain all static files and code may be organised into sub-directories such as CSS, JavaScript, etc)
  - there is a clear separation between custom code and any external files (for example, library files are all inside a directory named 'libraries')
  - files are named consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Readability
  - code is indented in a consistent manner to ease readability and there are no unnecessary repeated blank lines (and never more than 2)
  - id/class (CSS and JavaScript)/function/variable names clearly indicate their purpose
  - all code is split into well-defined and commented sections
  - semantic markup is used to structure HTML code
  - HTML, CSS, JavaScript and Python are kept in separate, linked files
  - CSS files are linked in the HTML file's head element
  - non-trivial JavaScript code files are linked at the bottom of the body element (or bottom of head element if needs loaded before the body HTML)
- Defensive design
  - all input data is validated (e.g. presence check, format check, range check)
  - internal errors are handled gracefully, and users are notified of the problem where appropriate.
- Comments
  - all custom code files include clear and relevant comments explaining the purpose of code segments
- Compliant code
  - HTML code passes through the official W3C validator with no issues
  - CSS code passes through the official (Jigsaw) validator with no issues
  - JavaScript code passes through a linter (e.g. jshint.com) with no major issues
  - Python code is consistent in style and conforms to the PEP8 style guide (or another explicitly mentioned style guide, such as Google's)
- Robust code
  - no logic errors are found when running code
  - errors caused by user actions are handled
  - where used, API calls that fail to execute or return data will be handled

gracefully, with the site users notified in an obvious way
    - inputs are validated when necessary
    - navigating between pages via the back/forward buttons can never break the site, there are no broken links
    - user actions do not cause internal errors on the page or in the console

The full design is implemented providing **a good solution to the users' demands and expectations** and **with consideration for security**.
Real world application

- Clearly understandable site-specific content is used rather than Lorem Ipsum placeholder text
- All links to external pages open in a separate tab when clicked
- The final application is aligned to the user stories presented at the start of the project

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. All noticeable errors have been corrected or documented:

Framework conventions are followed and used correctly.
Flask:

- Controllers
- Models
- Views
- Configuration and settings files are well-organised

Security features and practice are evidenced

- passwords and secret keys are stored in environment variables or in files that are in .gitignore, and are never committed to the repository
- any functionality requiring log-in is available only to logged-in users
- user permissions and levels of access are appropriate (e.g. a non-admin user would not be able to edit another user' post)

Data is well structured

- data is fully modelled and matches the schema.
- data store configuration is kept in a single location where it can be changed easily.
- data is well-structured
- all CRUD functionality is present and working and actions are immediately reflected in the front end

Configuration and dependencies files are kept up to date. Separate versions/branches of these are commits where relevant. Data store configuration is kept in a single location and can be changed easily. The data store is not accessible to the regular user without going through the code.

All noticeable errors have been corrected or documented:

- navigating between pages via the back/forward buttons can never break the site, there are no broken links
- user actions do not cause internal errors on the page or in the console

Version control software is used effectively:

- all code is managed in git with well-described commit messages

- there is a separate, well-defined commit for each individual feature/fix
- there are no very large commits which make it harder to understand the development process and could lead the assessor to suspect plagiarism

The full application development process is documented:

- the purpose of the application is clearly described in the README
- the project's documentation describes the UX design work undertaken for this project and the reasoning behind it
    - wireframes, mockups, diagrams, etc., created as part of the design process are included in the project
- there is a clear separation between code written by the learner and code from external sources (e.g. libraries or tutorials). All code from external sources is attributed to its source via comments above the code and (for larger dependencies) in the README.
- the data schema is clear, comprehensive, and easy to follow
- the data schema is fully documented in the README file
- A manual testing procedure is well documented either in the README or a separate file
- the deployment procedure is fully documented in a section in the README file
- clear, well-described, explanatory commit messages describe the development process
- the README is well-structured and easy to follow
- the README file is written in markdown and uses markdown formatting consistently and effectively
- project documentation and the application’s user interface have few errors in spelling and grammar

Unit 4: Full Stack Frameworks with Django

Level:	Level 6
Credit Value:	25
GLH:	190
Unit Number:	F/650/3528
Unit Aim:	This unit aims to provide learners with the knowledge and skills needed to build a Full Stack web application. The unit covers the use of frameworks; APIs; automated testing; persistent storage; user authentication and e-commerce functionality.

This unit has 5 learning outcomes.

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
1 Design, develop and implement a Full Stack web application, with a relational database, using the Django/Python Full Stack MVC framework and related contemporary technologies.	1.1 Design a Full Stack web application to be built using the Django framework and to incorporate a relational database and multiple apps (an app for each potentially reusable component). 1.2 Design a front end for a Full-Stack web application that meets accessibility guidelines, follows the principles of UX design, meets its given purpose and provide a set of user interactions. 1.3 Develop and implement a Full Stack web application built using the Django framework, to incorporate a relational	M(i) Design and build a real-world Full Stack MVC application with a Front end: - that is easy to navigate and allows the user to find information and resources intuitively. - where the user has full control of their interaction with the application. - where all data (CRUD) actions are immediately reflected in the user interface. - where the purpose is immediately evident to a	**Characteristics of Performance at Distinction

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	database, an interactive front-end and multiple apps (an app for each potentially reusable component). 1.4 Implement at least one form, with validation, that allows users to create and edit models in the Back end. 1.5 Build a Django file structure that is consistent and logical, following the Django conventions. 1.6 Write code that clearly demonstrates characteristics of 'clean code'. 1.7 Define application URLs in a consistent manner. 1.8 Incorporate a main navigation menu and structured layout. 1.9 Include custom logic that shows proficiency in the Python language. 1.10 Write Python code that includes functions with compound statements such as if conditions and/or loops. 1.11 Design and implement manual or automated test procedures to assess functionality, usability, responsiveness and data	new user. - which provides a good solution to the user's demands and expectations. - which has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences). M(ii) Produce a robust codebase M(iii) Follow a Test Driven Development (TDD) approach (for JavaScript and/or Python), demonstrated in the git commits. M(iv) Configure the project efficiently through well-kept Procfile, requirements.txt file, settings files, keeping the data store configuration in a single location where it can be changed easily.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	management within the full web application.		
2 Design and implement a relational data model, application features and business logic to manage, query and manipulate relational data to meet given needs in a particular real-world domain.	2.1 Design a relational database schema with clear relationships between entities. 2.2 Create at least TWO original custom Django models. 2.3 Create at least one form with validation that will allow users to create records in the database (in addition to the authentication mechanism). 2.4 Implement all CRUD (create, select/read, update, delete) functionality	M(v) Fully describe the data schema in the project README file.	
3 Identify and apply authorisation, authentication and permission features in a full-stack web application solution.	3.1 Implement an authentication mechanism, allowing a user to register and log in, stipulating a clear reason as to why the users would need to do so. 3.2 Implement log-in and registration pages that are only available to anonymous users. 3.3 Implement functionality that prevents non-admin users from accessing the data store directly without going through the code.	M(vi) Demonstrate solid understanding of Django template syntax, logic and usage, placing Django logic in the component where it is best suited (e.g., data handling logic is in the models, business logic is in the views).	
4 Design, develop and integrate an e-commerce payment system in a cloud-hosted Full Stack web application.	4.1 Implement at least one Django app containing some e-commerce functionality using an online payment processing		

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	system (e.g. Stripe). This may be a shopping cart checkout, subscription-based payments or single payments, donations, etc. 4.2 Implement a feedback system that reports successful and unsuccessful purchases to the user, with a helpful message.		
5 Document the development process through a git based version control system and deploy the full application to a cloud hosting platform.	5.1 Deploy the final version of your code to a hosting platform and test that it matches the development version. 5.2 Ensure that all final deployed code is free of commented out code and has no broken internal links. 5.3 Ensure the security of the deployed version, making sure to not include any passwords in the git repository, that all secret keys are hidden in environment variables or in files that are in gitignore, and that DEBUG mode is turned off. 5.4 Use a git-based version control system for the full application, generating documentation through regular commits and in the project README.	M(vii) Use version control software effectively to provide a record of the development process.	

LEARNING OUTCOMES The learner will:	ASSESSMENT CRITERIA - PASS The learner can:	MERIT CRITERIA* In addition to the pass criteria, the learner can:	DISTINCTION
	5.5 Create a project README that is well-structured and written using a consistent markdown format. 5.6 Document the full deployment procedure, including the database, and the testing procedure, in a README file that also explains the application's purpose and the value that it provides to its users.		

*Additional guidance for merit

To achieve merit learners, need to meet the assessment criteria outlined above for a pass and a merit.

The following additional guidance describes characteristics of performance at MERIT.

The learner has a clear rationale for the development of this project and has produced a fully functioning, well-documented, relational database backed, Full Stack application for a real-life audience, with a full set of CRUD creation, reading, updating and deletion of data records) features . Data validation, API handling and user feedback are all evident in the code and the working application. Templates have been used correctly produce working features. There are no logic errors in the code and the application functions as expected.

The finished project has a clear, well-defined purpose addressing the needs of a particular target audience (or multiple related audiences) and a particular data domain. Its purpose would be immediately evident to a new user without having to look at supporting documentation. The user is kept informed of progress and actions through feedback and, where large data sets are being loaded, progress indicators.
The design of the web application follows the principles of UX design and accessibility guidelines, and the site is fully responsive.

Data is fully modelled and matches the schema. The schema design is documented in the README. Data store configuration is kept in a single location and can be changed easily. Configuration and settings files are well-organised and there are different versions for different branches.

Code is well-organised and easy to follow, and the application has been fully tested, following a test-driven development approach and a planned, testing procedure, with no obvious errors left in the code.

The development process is clearly evident through detailed commit messages. The project's documentation provides a clear rationale for the development of this project and covers all stages of the development life cycle.

The application is robust and deals with external errors gracefully (user input, API calls, asynchronous processes).

The conventions of the framework are followed, and code is clearly separated with HTML, CSS, JavaScript and Python being kept in separate, linked files.

All logic in the app makes sense in the context of its purpose (e.g. if different users have different permissions, then their access levels are appropriate so that, for example, a non-admin user will not be able to edit another user's data.

**Characteristics of Performance at Distinction

To achieve a distinction, a learner will have achieved all pass and merit criteria, as described above, and will demonstrate characteristics of high-level performance as described below:

Characteristics of performance at distinction:

The learner has documented a clear, justified, rationale for a real-world application and a comprehensive explanation of how it will be developed. The development of the project has

resulted in a fully-functioning, interactive, Full Stack Django application, with well-designed data and a full set of CRUD operations. The learner shows a clear understanding of data modelling techniques and of the relationship between the Back end and Front end.

The finished project is judged to be publishable in its current form with a professional grade user interface and functionality, and interaction adhering to current practice. There are no logic errors in the code. Where there is a clear breach of accepted design/UX principles, or of accepted good practice in code organisation, these are fully justified, appropriate and acceptable to the target user. It clearly matches the design and demonstrates the characteristics of craftsmanship in the code. The database schema is representative of complex user stories and there is a fully documented and full set of data operations which are fit for purpose in relation to the domain.

Each individual app matches a natural aspect of the project, with no app being too small or too big (a common approach is to have an app to encapsulate each single set of tightly connected models). Any data in the models that is relevant to multiple apps is shared, rather than duplicated.

The resulting application is original and not a copy of any walkthrough projects encountered in the unit.

Amplification (craftsmanship)this amplification is only applicable to performance at distinction.

Front end Design

The design of the web application demonstrates the main principles of good UX design:

- Information Hierarchy
  - semantic markup is used to convey structure - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all information displayed on the site is presented in an organised fashion with each piece of information being easy to find
  - all resources on the site are easy to find, allowing users to navigate the layout of the site intuitively
  - information is presented and categorised in terms of its priority
- User Control
  - all interaction with the site would be likely to produce a positive emotional response within the user. This is down to the flow of information layout, use of colour, clear and unambiguous navigation structures and all interaction feedback
  - when displaying media files, the site avoids aggressive automatic pop-ups and autoplay of audio; instead letting the user initiate and control such actions
  - users who direct to a non-existent page or resource are redirected back to the main page without having to use the browser navigation buttons
  - users are never asked for information that the application already has (e.g. a contact form does not ask a logged in user for an email address).
  - the user is shown progress indicators and feedback on transactions.
  - errors resulting from user or data actions are reported to the user
- Consistency
  - evident across all pages/sections and covers interactivity as well as design
  - consistency across all data operations, including in the reporting
- Confirmation
  - user and data actions are confirmed where appropriate, feedback is given at all times
- Accessibility
  - there is clear conformity to accessibility guidelines across all pages/sections and in all interactivity

Any design decisions that contravene accepted user interaction, user experience design principles are identified and described (comments in code and/or a section in the README)

Development and Implementation

Code demonstrates characteristics of ‘clean code’:

- Consistent and appropriate naming conventions within code and in file naming, e.g.
  - file names, class names, function names and variable names are descriptive and consistent

    - for cross-platform compatibility, file and directory names will not have spaces in them and will be lower-case only
    - all HTML attributes, CSS rules, code variables and function names are consistent in format, follow standards for the language and are appropriate and meaningful
    - app urls are consistent
- File structure
    - whenever relevant, files are grouped in directories by file type (e.g. an assets directory will contain all static files and code may be organised into sub-directories such as CSS, JavaScript, etc)
    - there is a clear separation between custom code and any external files (for example, library files are all inside a directory named 'libraries')
    - files are named consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.
- Readability
    - code is indented in a consistent manner to ease readability and there are no unnecessary repeated blank lines (and never more than 2)
    - id/class(CSS and JavaScript)/function/variable names clearly indicate their purpose
    - all code is split into well-defined and commented sections
    - Semantic markup is used to structure HTML code
    - HTML, CSS, JavaScript and Python are kept in separate, linked files
    - CSS files are linked in the HTML file's head element
    - non-trivial JavaScript code files are linked to at the bottom of the body element (or bottom of head element if it needs to be loaded before the body HTML)
- Defensive design
    - all input data is validated (e.g. presence check, format check, range check)
    - internal errors are handled gracefully, and users are notified of the problem where appropriate
- Comments
    - all custom code files include clear and relevant comments explaining the purpose of code segments
- Compliant code
    - HTML code passes through the official W3C validator with no issues
    - CSS code passes through the official (Jigsaw) validator with no issues
    - JavaScript code passes through a linter (e.g. jshint.com) with no major issues
    - Python code is consistent in style and conforms to the PEP8 style guide (or another explicitly mentioned style guide, such as Google's)
- Robust code
    - no logic errors are found when running code
    - errors caused by user actions are handled
    - where used, API calls that fail to execute or return data will be handled gracefully, with the site users notified in an obvious way
    - inputs are validated when necessary
    - navigating between pages via the back/forward buttons can never break the site, there are no broken links
    - user actions do not cause internal errors on the page or in the console

The full design is implemented providing **a good solution to the users' demands and expectations** and **with consideration for security**.

Real world application

- Clearly understandable site-specific content is used rather than Lorem Ipsum placeholder text
- All links to external pages open in a separate tab when clicked
- The final application is aligned to the user stories presented at the start of the project

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. All noticeable errors have been corrected or documented:

Security features and practice are evidenced

- passwords and secret keys are stored in environment variables or in files that are in .gitignore, and are never committed to the repository
- any functionality requiring log-in is available only to logged-in users
- user permissions and levels of access are appropriate (e.g. a non-admin user would not be able to edit another user' post)

Framework conventions are followed and used correctly. Including the following:

Django/Flask:

- Templates
- Apps
- Models
- Views
- Placing of logic in the most appropriate components demonstrates an understanding of the Model-View-Controller (Template) pattern is evident through the placing of logic in the most appropriate components.
- Configuration and settings files are well-organised

Security features and practice are evidenced

- passwords and secret keys are stored in environment variables or in files that are in .gitignore, and are never committed to the repository
- any functionality requiring log-in is available only to logged-in users
- user permissions and levels of access are appropriate (e.g. a non-admin user would not be able to edit another user' post)

Data is well structured

- data is fully modelled and matches the schema.
- data store configuration is kept in a single location where it can be changed easily.
- data is well-structured, organised into logical entities with clear relationships between them
- all CRUD functionality is present and working and actions are immediately reflected in the front end
- any data used across multiple apps is shared and not duplicated.

Configuration and dependencies files are kept up to date. Separate versions/branches of these are commits where relevant. Data store configuration is kept in a single location and can be changed easily. The data store is not accessible to the regular user without going through the code.

Testing procedures are comprehensive, with a good level of coverage, and have clearly been followed. There is clear evidence of a test-driven development approach and this is demonstrated in git commits. All noticeable errors have been corrected or documented:

- navigating between pages via the back/forward buttons can never break the site, there are no broken links
- user actions do not cause internal errors on the page or in the console

Version control software is used effectively:

- all code is managed in git with well-described commit messages
- there is a separate, well-defined commit for each individual feature/fix
- there are no very large commits which make it harder to understand the development process and could lead the assessor to suspect plagiarism
- there are different versions of configuration and settings files for different branches

The full application development process is documented:

- the purpose of the application is clearly described in the README
- the project's documentation describes the UX design work undertaken for this project and the reasoning behind it
    - wireframes, mock-ups, diagrams, etc., created as part of the design process are included in the project
- there is a clear separation between code written by the learner and code from external sources (e.g. libraries or tutorials). All code from external sources is attributed to its source via comments above the code and (for larger dependencies) in the README
- the data schema is clear, comprehensive, and easy to follow.
- the data schema is fully documented in the README file
- the testing procedure is well documented either in the README or a separate file
- the deployment procedure is fully documented in a section in the README file
- clear, well-described, explanatory commit messages describe the development process
- the README is well-structured and easy to follow
- the README file is written in markdown and uses markdown formatting consistently and effectively
- project documentation and the application’s user interface have few errors in spelling and grammar

gateway qualifications

enquiries@gatewayqualifications.org.uk
www.gatewayqualifications.org.uk
Tel: 01206 911 211

Charity Registration No. 114282
Registered in England Company No. 5502449