



ICT171 WEB SERVER DOCUMENTATION

Student: Bhuvanesh Krishnan Vijayaraj (34580045)

Project Name: Ctrl + Shift + Game

GitHub Repository: <https://github.com/Bhuvanesh-Murdoch2005/ict171-assignment2>

Table of Contents

1.0 Web Server Setup In AWS	4
1.1 EC2 Setup	4
1.2 Accessing AWS and Navigating to EC2	4
1.3 Launching a new EC2 Instance	5
1.4 Security Group Configuration	5
1.5 Final Launch	6
1.6 SSH Key Setup and Terminal Access	6
2.0 Apache Installation and Configuration	7
2.1 Updating Package Lists	7
2.2 Installing Apache	7
2.3 Starting and Enabling Apache and	7
2.4 Verifying Apache Status	8
.....	8
2.5 Apache Hosting Test	8
3.0 Domain Setup and SSL	9
3.1 Domain Name Configuration	9
3.2 SSL Certificate with Let's Encrypt (Certbot)	10
3.3 DNS and SSL/TLS Documentation	11
4.0 GitHub Progress and Version Control	12
4.1 Account Creation	12
4.2 Repository Setup	12
4.3 Commit History	13
4.4 Pushing Code and Updates	14
4.5 GitHub Pages or Actions	15
4.6 Branching Strategy	16
6.0 Docker Installation and Account Setup	17
6.1 Docker Installation on MacBook Air (Apple M2) and Initial Setup	17
6.2 Docker Compose Configuration	20
6.3 Docker Deployment Scripts	21
6.3.1 Dockerfile Script	21
6.3.2 Automation Script (setup.sh)	21
7.0 Final Testing and Verification	24
7.1 Web Application Access and Security	24
7.1.1 Website Accessibility via DNS	24
7.1.2 SSL/TLS Functionality	24
7.1.3 Docker Container Persistence	24
7.1.4 Page Routing and Functional Testing	25

7.2 Final Live Website Deployment Screenshot.....	26
7.3 Troubleshooting Tips	30
7.3.1 Apache Not Reflecting Updated Files	30
7.3.2 Docker Port Conflict	30
7.3.3 SSL Certificate Not Valid After Setup	31
7.3.4 Website Doesn't Load After EC2 Reboot.....	31
7.3.5 Files Not Loading Inside Container	31
7.3.6 Certbot Not Installing SSL	32
8.0 Licensing	32

1.0 Web Server Setup In AWS

1.1 EC2 Setup

I hosted on Amazon EC2 (Elastic Compute Cloud) to construct and set up a secure Ubuntu-based server for my static blog website. This configuration is necessary to make sure that my website can be accessed, maintained, and grown on a cloud platform.

1.2 Accessing AWS and Navigating to EC2

To begin, I signed in to the [AWS Management Console](#). If you do not have an account yet, you can create one by clicking on the “**Sign in to the Console**” button on the AWS homepage and then selecting “**Create a new AWS account**”.

Once logged in, I searched for **EC2** in the search bar at the top of the console to access the EC2 dashboard.

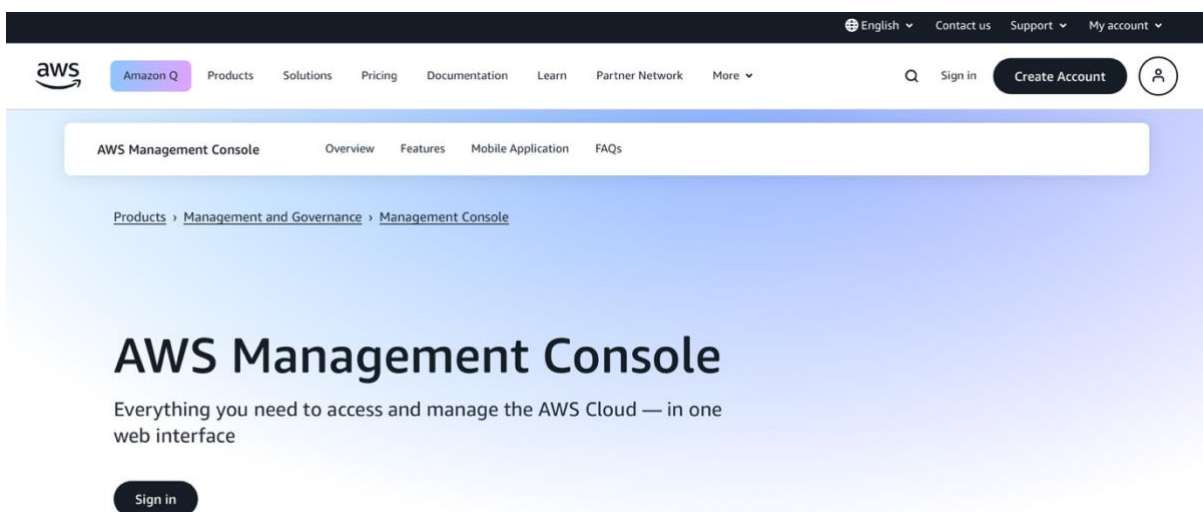


Figure 1: AWS Homepage – Starting point for signing in or creating an account before launching EC2 services.

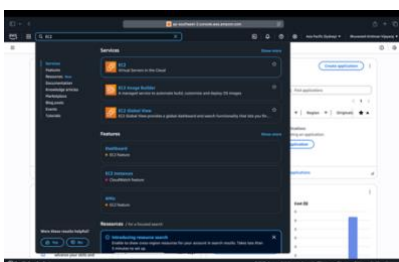


Figure 2: Searching for EC2 under AWS services to begin configuring a virtual server instance.

1.3 Launching a new EC2 Instance

I clicked on "**Launch Instance**" to begin configuring a new virtual server.

- **Name:** CtrlShiftGame
- **Application and OS Images**
AMI: Ubuntu Server 24.04 LTS
- **Instance Type:** t2.micro (Free tier eligible)

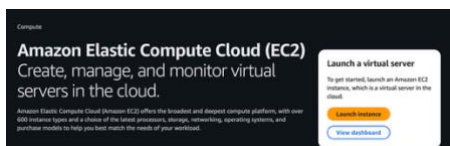


Figure 3: EC2 overview page on AWS, providing options to launch and manage virtual server instances in the cloud.

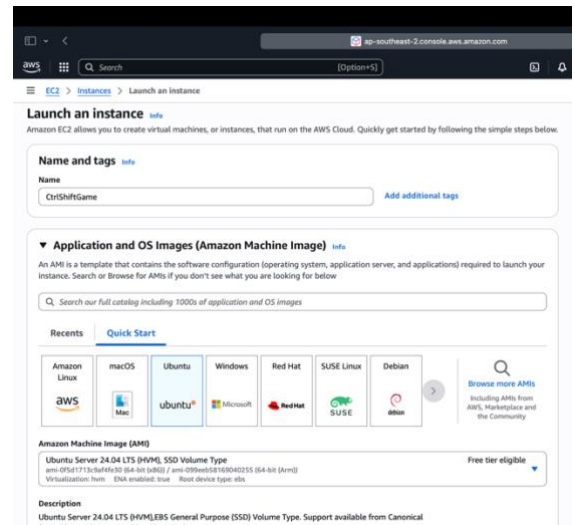


Figure 4: Instance name and AMI selection

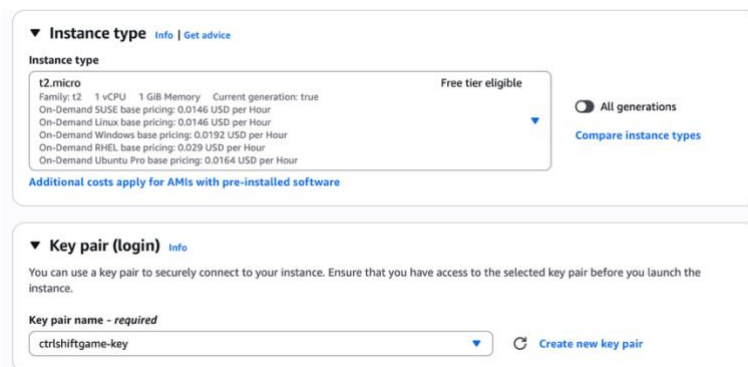


Figure 5: Selecting t2.micro instance type and attaching existing key pair.

1.4 Security Group Configuration

I created a new security group with an inbound rule for **SSH (port 22)** access from all IP addresses to allow remote connection to the instance.

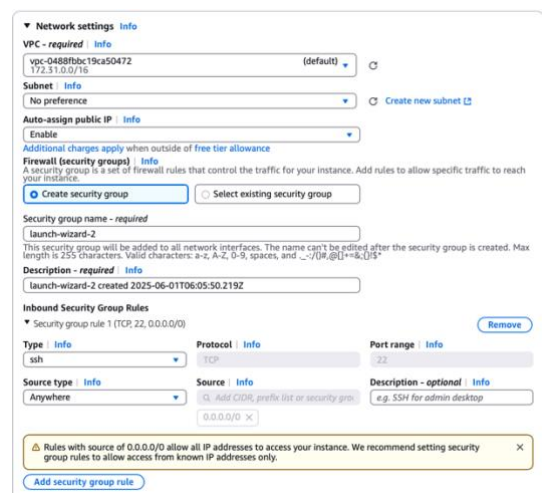


Figure 6: Firewall and network settings configured to allow SSH.

1.5 Final Launch

After reviewing the setup, I clicked “**Launch Instance**” to deploy the server.

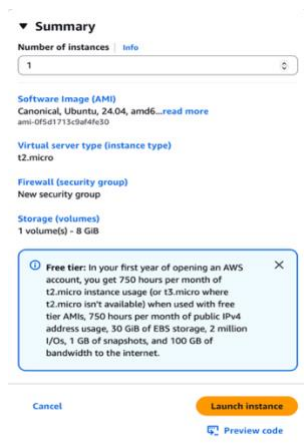


Figure 7: Summary page showing AMI, instance type, and firewall config.

1.6 SSH Key Setup and Terminal Access

Once the instance was running, I connected using the terminal from my Mac device with the .pem key file using this command:

```
Last login: Sun Jun  1 19:43:28 on ttys000
(base) bhuvanesh@MacBook-Air-71 ~ % cd ~/Downloads

(base) bhuvanesh@MacBook-Air-71 Downloads % chmod 400 ctrlshiftgame-key.pem

[(base) bhuvanesh@MacBook-Air-71 Downloads % ssh -i "ctrlshiftgame-key.pem" ubuntu@54.66.59.90]
```

Figure 8: Terminal commands

Upon successful connection:

- OS Verified: Ubuntu 24.04.2 LTS
- IP Address: 54.66.59.90 (Internal)

```
(base) bhuvanesh@MacBook-Air-71 ~ % cd ~/Downloads
(base) bhuvanesh@MacBook-Air-71 Downloads % chmod 400 ctrlshiftgame-key.pem
[(base) bhuvanesh@MacBook-Air-71 Downloads % ssh -i "ctrlshiftgame-key.pem" ubuntu@54.66.59.90]
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sun Jun  1 06:12:47 UTC 2025

System load:  0.0               Processes:    109
Usage of /:   42.1% of 6.71GB   Users logged in: 0
Memory usage: 35%              IPv4 address for enX0: 172.31.14.165
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.
   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

46 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***
Last login: Wed May 28 10:46:01 2025 from 134.115.176.166
ubuntu@ip-172-31-14-165:~$
```

Figure 9: Terminal login via SSH to the newly created EC2 instance.

2.0 Apache Installation and Configuration

To host my website on the cloud, I installed the **Apache HTTP Server** on my Ubuntu-based EC2 instance. Apache is a reliable and widely used open-source web server that enables HTTP communication between the browser and the server. The installation was done through the terminal using simple package management commands.

2.1 Updating Package Lists

Before installing any new software, I updated the package lists to ensure all repositories were current.

```
ubuntu@ip-172-31-14-165:~$ sudo apt update
```

Figure 10: Updating Ubuntu package repositories on the EC2 instance.

2.2 Installing Apache

After updating the package index, I installed Apache with the following command:

```
ubuntu@ip-172-31-14-165:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Figure 11: Apache installation confirmation showing apache2 is already the newest version or freshly installed.

2.3 Starting and Enabling Apache and

To ensure Apache runs after every reboot, I used the following commands:

```
ubuntu@ip-172-31-14-165:~$ sudo systemctl start apache2
ubuntu@ip-172-31-14-165:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
```

Figure 12: Starting and enabling the Apache service on the instance.

2.4 Verifying Apache Status

```
ubuntu@ip-172-31-14-165:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-06-01 07:54:40 UTC; 4h 44min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 192823 (apache2)
    Tasks: 55 (limit: 1129)
  Memory: 18.7M (peak: 22.8M)
    CPU: 1.204s
   CGroup: /system.slice/apache2.service
           └─192823 /usr/sbin/apache2 -k start
             └─192825 /usr/sbin/apache2 -k start
               └─192826 /usr/sbin/apache2 -k start
```

Figure 13: Apache status check confirming the web server is running and enabled.

2.5 Apache Hosting Test

After Apache was successfully installed and running, I tested my deployed website by visiting the EC2 public IP address in a browser:



Figure14: Website homepage loaded using Apache server on public EC2 IP.

My website loaded successfully and showed the landing page

3.0 Domain Setup and SSL

After setting up the server and hosting the static website, I linked a custom domain name to make the website more professional and accessible. I also implemented SSL encryption to ensure secure communication between users and the server. Troubleshooting tips: If `sudo certbot` doesn't work, ensure Apache is running and port 80 is open

3.1 Domain Name Configuration

I purchased the domain `bhuvi.xyz` from GoDaddy (<https://www.godaddy.com/en-au>) and configured it to point to my EC2 server's public IP address.

1. Logged into [GoDaddy](https://www.godaddy.com) and navigated to the **DNS Management** page (Figure 15).
2. Created an A record:
 - **Type:** A
 - **Name:** @
 - **Value:** 54.66.59.90 (EC2 Public IPv4)
 - **TTL:** ½ Hour (Figure 2)

This step ensured that when someone types `bhuvi.xyz` in their browser, it routes to the hosted EC2 instance (Figure 17). As well as DNS propagation may take up to 30 minutes. You can check progress at <https://dnschecker.org>

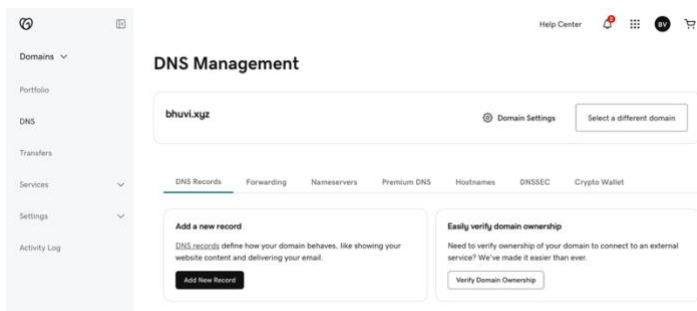


Figure 15: GoDaddy domain dashboard showing the registered domain used to link with the cloud-hosted website.

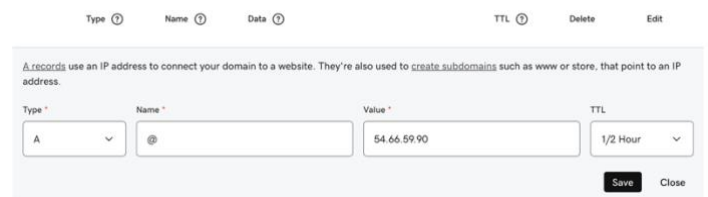


Figure 16: GoDaddy A Record configuration pointing `bhuvi.xyz` to the EC2 instance IP address 54.66.59.90. The TTL (Time

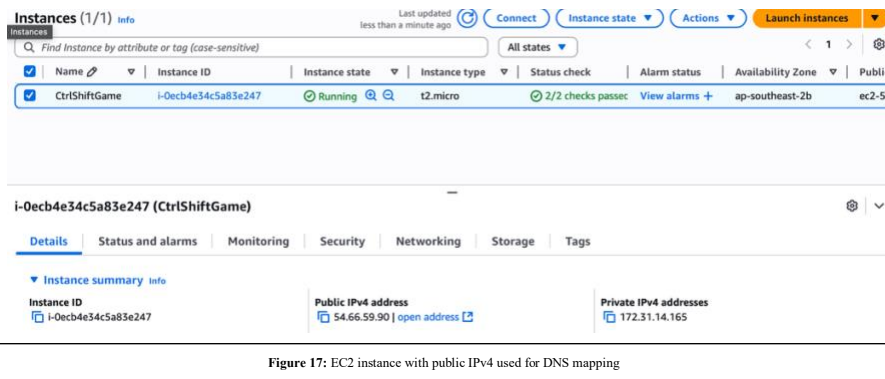


Figure 17: EC2 instance with public IPv4 used for DNS mapping

After DNS propagation completed, I verified the domain connection by loading my hosted webpage at <http://bhuvi.xyz>. The static site loaded correctly, confirming the domain was successfully linked (Figure 18).

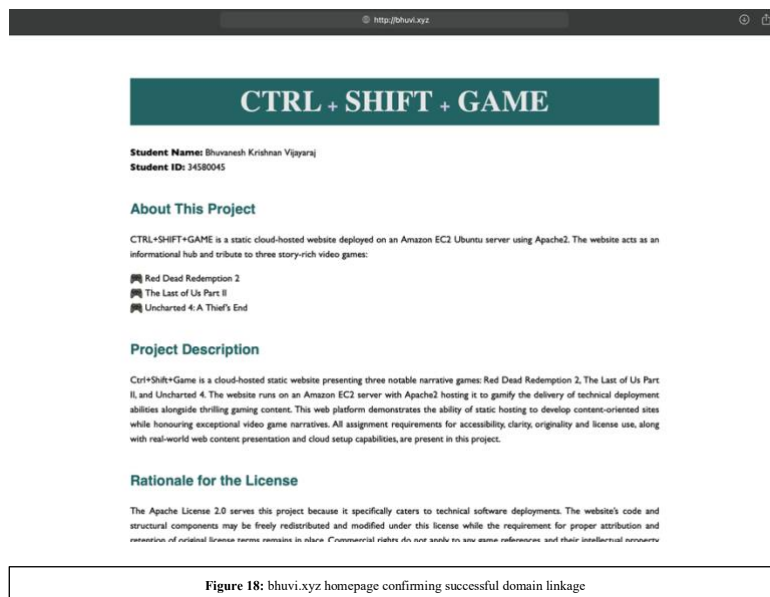


Figure 18: bhuvi.xyz homepage confirming successful domain linkage

3.2 SSL Certificate with Let's Encrypt (Certbot)

To enable HTTPS, I used **Certbot** by **Let's Encrypt** to install a free SSL certificate on the server.

- Ran `sudo certbot --apache` to install the certificate.
- Verified the certificate with `sudo certbot certificates` (Figure 19), which shows:
 - Certificate Name: bhuvi.xyz
 - Expiry Date: 2025-08-24
 - Certificate and private key paths

```

ubuntu@ip-172-31-14-165:~$ sudo certbot certificates
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-----
Found the following certs:
Certificate Name: bhuvi.xyz
Serial Number: 559cff29191cb09cc74b1db7142604b91a5
Key Type: ECDSA
Domains: bhuvi.xyz www.bhuvi.xyz
Expiry Date: 2025-08-24 13:58:07+00:00 (VALID: 84 days)
Certificate Path: /etc/letsencrypt/live/bhuvi.xyz/fullchain.pem
Private Key Path: /etc/letsencrypt/live/bhuvi.xyz/privkey.pem
-----

```

Figure 19: Terminal output confirming SSL certificate issuance for bhuvi.xyz

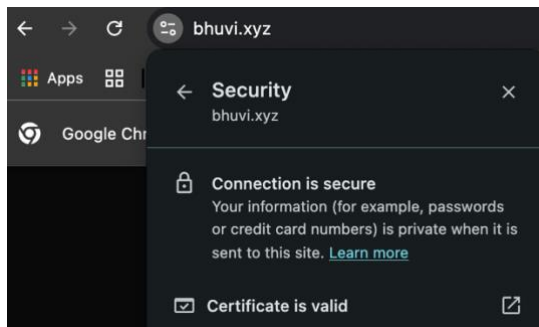


Figure 20: Browser showing secure connection and valid SSL certificate for the domain

3.3 DNS and SSL/TLS Documentation

To secure and expose my website publicly, I configured DNS and SSL as follows:

- **DNS Configuration:**

I linked my domain bhuvi.xyz (purchased via GoDaddy) to the public IP address of my AWS EC2 instance by updating the A record.

- Refer to **Figure 15** for DNS configuration on GoDaddy.

- **SSL/TLS Certificate Setup:**

I used **Let's Encrypt Certbot** to issue and install a valid SSL certificate for my domain. This ensures **secure HTTPS access**, improves trust, and meets modern web standards.

- Refer to **Figure 19** showing the successful certificate installation via Certbot.

- **Verification:**

After configuration, accessing <https://bhuvi.xyz> loaded my Apache-hosted website securely with a valid certificate (green padlock icon).
- Refer to **Figure 20** for the browser HTTPS padlock and the terminal Certbot success.

4.0 GitHub Progress and Version Control

4.1 Account Creation

As a Murdoch University student, I registered for [GitHub](#) using my personal details and selected Australia as my region. I also applied for the [GitHub Student Developer Pack](#) to access premium tools like GitHub Copilot and free repository hosting (Figure 1).

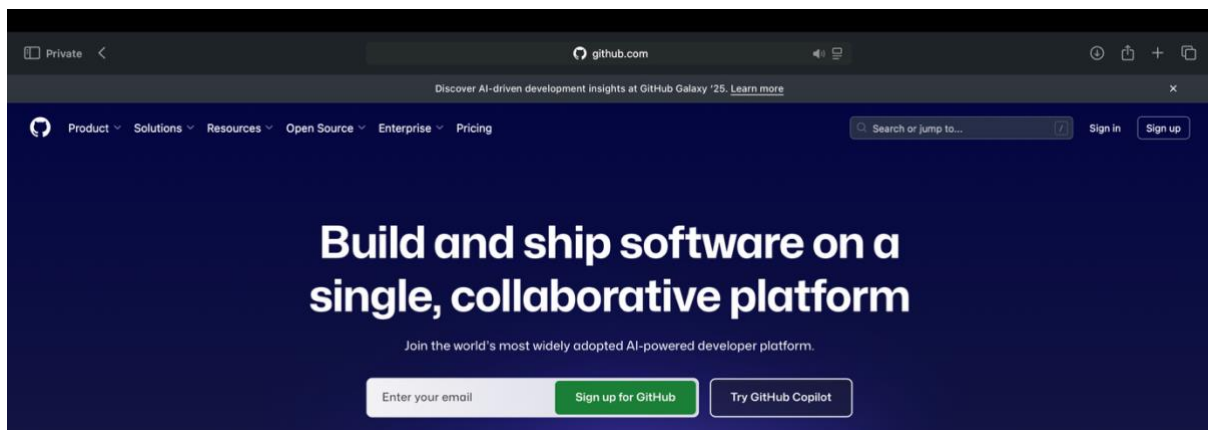


Figure 21: GitHub Account Setup (GitHub homepage)

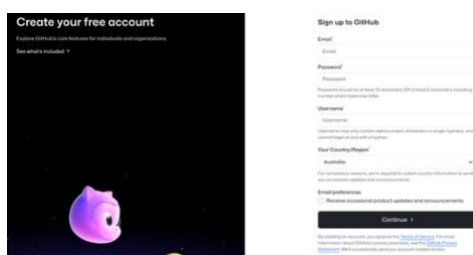


Figure 22: GitHub Account Setup (Sign-up screen)



Figure 23: GitHub Account Setup (Student developer pack page)

4.2 Repository Setup

I created a new public repository named `ict171-assignment2` and included a README file to introduce the project (Figure 24). The repo is titled “ctrl-shift-game” and represents my gaming blog hosted on EC2 (Figure 25).

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * Bhuvanesh-Murdoch2005 / Repository name * ict171-assignment2.
✔ ict171-assignment2 is available.

Great repository names are short and memorable. Need inspiration? How about [laughing-computing-machine](#)?

Description (optional)

☒ **Public**
 Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
 You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
 This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore
 .gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license
 License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

[Create repository](#)

Figure 24: Repository creation page showing repo name ict171-assignment2, public visibility, and README initialisation

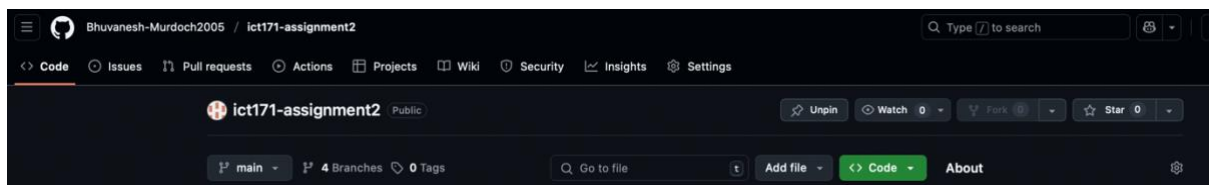


Figure 25: GitHub repository dashboard for ict171-assignment2 under user Bhuvanesh-Murdoch2005. This view shows the repository is public, the active branch is main, and a total of 4 feature branches have been created.

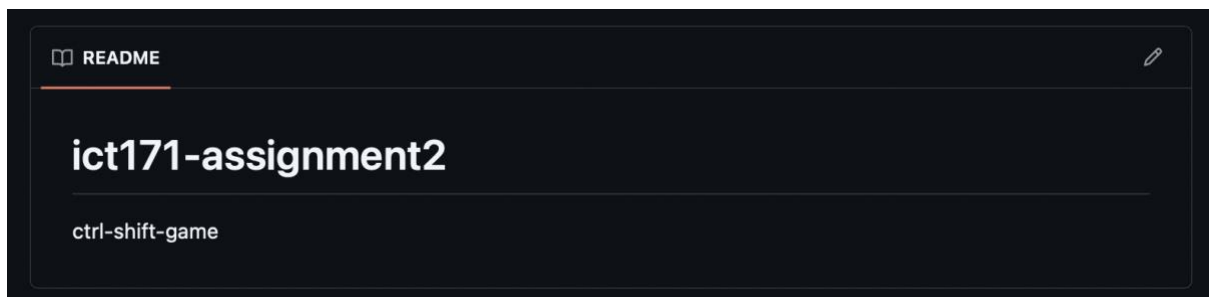


Figure 26: README.md section of the ict171-assignment2 repository. Displays the repository title and short description "ctrl-shift-game", confirming initialization.

4.3 Commit History

Multiple commits were made to track progressive updates, such as:

- Initial commit
- UI updates
- Dockerfile creation

- Docker Compose setup
(Figure 4 shows the commit timeline in GitHub.)

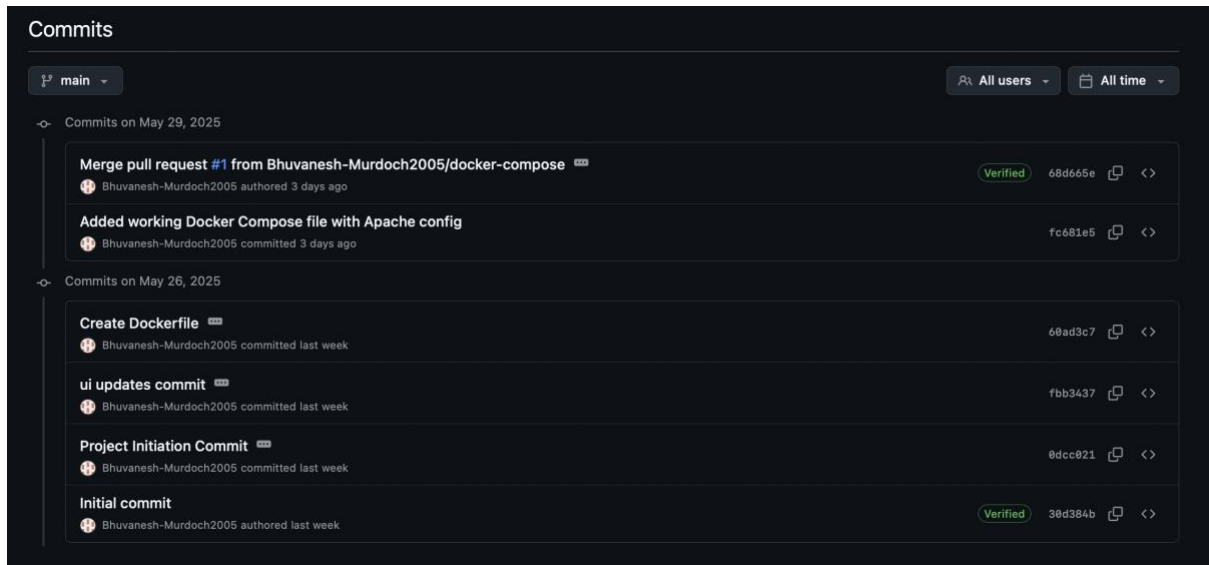


Figure 27: GitHub repository main page after creating ict171-assignment2 with main as the default branch.

4.4 Pushing Code and Updates

I used the terminal to push local changes to GitHub using the `git push -u origin main` command after resolving sync issues via `git pull --allow-unrelated-histories` (Figure 28).

Recent pushes included key files like `index.html`, `Dockerfile`, and `docker-compose.yml`, as shown in the commit section of the repo.

```
(base) bhuvanesh@MacBook-Air-71 push-demo-test % git remote add origin https://github.com/Bhuvanesh-Murdoch2005/push-demo-test.git

(base) bhuvanesh@MacBook-Air-71 push-demo-test % git push -u origin main

Username for 'https://github.com': ghp_CaLx9AXJXVSm1RXqTHxNC1ct5ysyL515149X
Password for 'https://ghp_CaLx9AXJXVSm1RXqTHxNC1ct5ysyL515149X@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 252 bytes | 252.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Bhuvanesh-Murdoch2005/push-demo-test.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) bhuvanesh@MacBook-Air-71 push-demo-test %
```

Figure 28: Terminal screenshot confirming `git push -u origin main` success using a personal access token.

4.5 GitHub Pages or Actions

While GitHub Pages deployment was not used in this project, I visited the Pages section to review options (Figure 29).

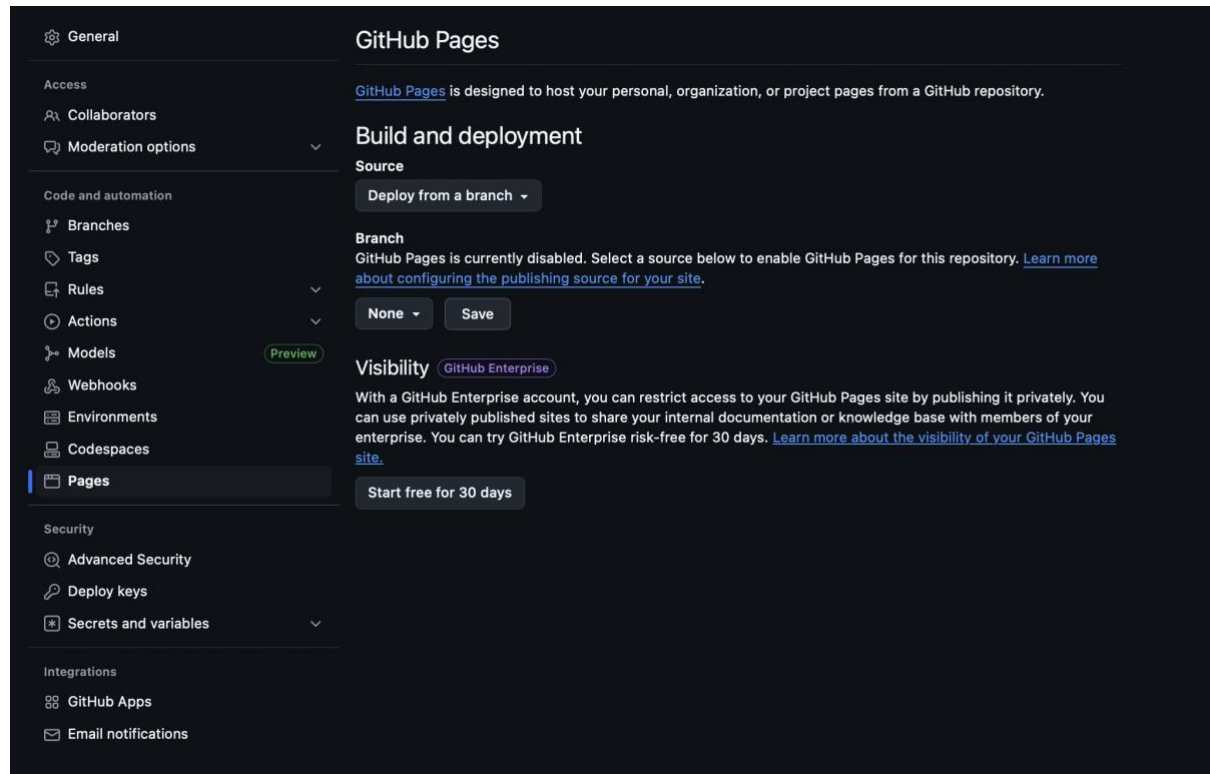


Figure 29: GitHub Pages settings showing deployment is currently disabled but available under the "Pages" section.

Under GitHub Actions, Docker-based workflows were suggested for continuous integration. These were explored but not configured, since deployment was done manually via EC2 (Figure 30).

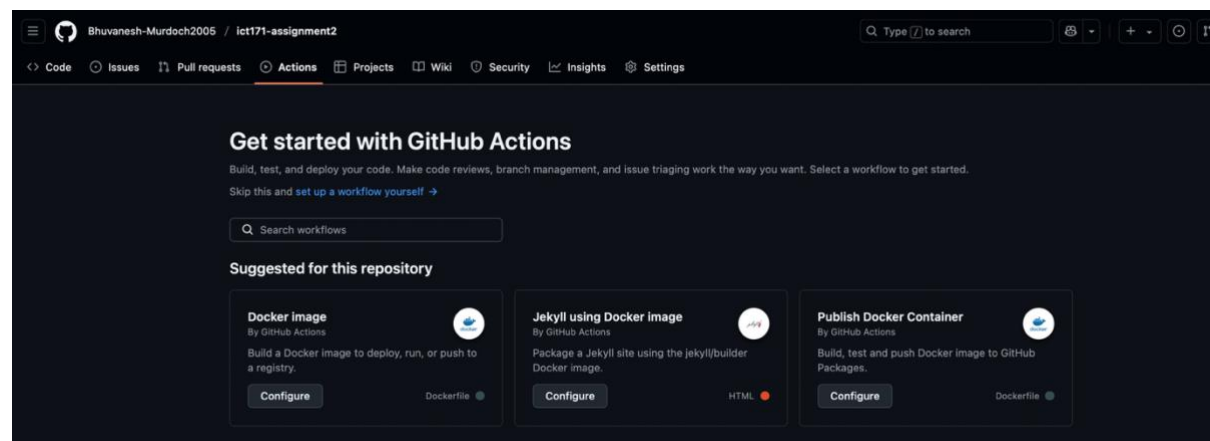


Figure 30: GitHub Actions tab showing suggested Docker-related workflows available for setup.

4.6 Branching Strategy

To support modular development, I created multiple branches:

- static-blog-site for initial HTML structure
 - ui-feature for user interface styling
 - dockerfile-setup for containerization
- (Figures 8–10 show the creation of each branch.)

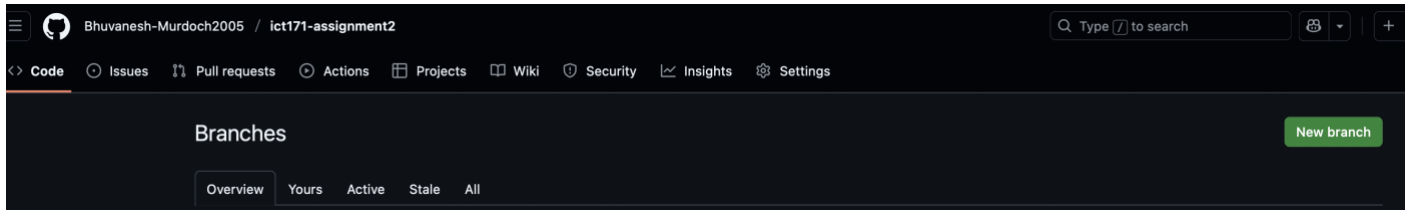


Figure 31: GitHub Branches section showing all available branches created for iterative feature development within the ict171-assignment2 repository.

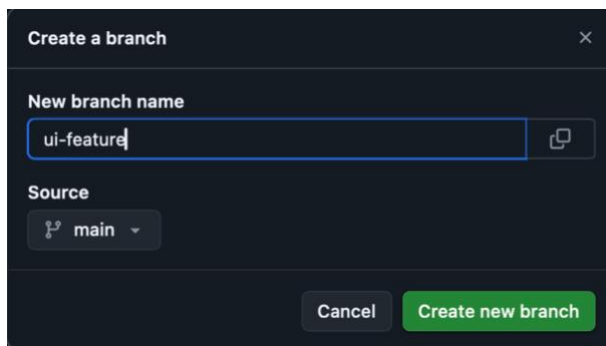


Figure 32: Creating a new branch named ui-feature from the main branch to implement user interface improvements separately from the production code.

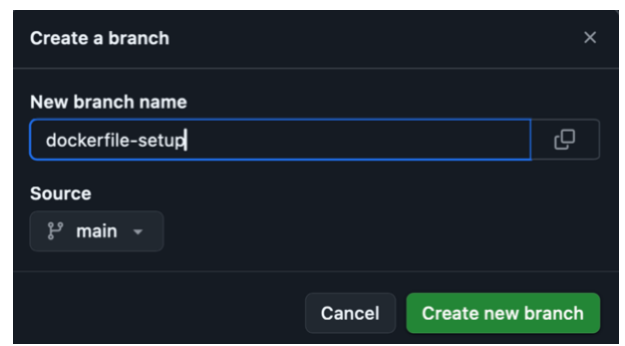


Figure 33: Creating a new branch named dockerfile-setup from the main branch to isolate development and testing of the Dockerfile configuration.

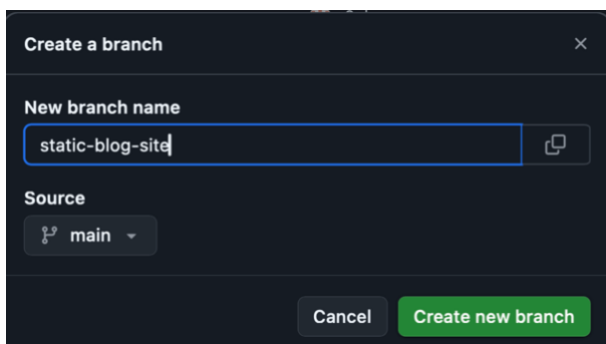


Figure 34: Creating a new branch named static-blog-site from the main branch to develop the static HTML version of the gaming blog separately.

6.0 Docker Installation and Account Setup

6.1 Docker Installation on MacBook Air (Apple M2) and Initial Setup

To enable **containerised deployment** of my web server project, I installed **Docker Desktop** on my MacBook Air (Apple M2). Docker is a widely used platform that simplifies the process of building, sharing, and running applications within lightweight, portable containers. Using Docker ensures that my application can be deployed consistently across different environments without the typical "it works on my machine" issues.

By containerising my Ctrl + Shift + Game website, I was able to:

- **Package the full stack (HTML, CSS, Apache server) into a single unit**, making it easy to ship and run.
- **Isolate dependencies**, reducing the risk of version conflicts or misconfiguration.
- **Ensure consistent performance** on both local machines and cloud servers like AWS EC2.
- **Use version control and automation tools** with better integration (e.g., Docker Compose and GitHub Actions in later steps).

Here's how I set up Docker:

1. Downloading Docker Desktop

I accessed the official Docker website (<https://www.docker.com>) and clicked on the **“Download Docker Desktop”** button for macOS. Docker provides native support for Apple Silicon (M1/M2), which ensures optimal performance and compatibility with my device.



Figure 35: Docker homepage



Figure 36: Accessing the official Docker website to begin the installation process for Docker Desktop on macOS (Apple M2).

2. Installing Docker

After downloading the installer, I followed the macOS installation prompts. The Docker Desktop application was then successfully installed and launched on my machine.

3. Creating a Docker Account & Logging In

Upon launching Docker Desktop, I **signed into my Docker Hub account**. Having a Docker account is important as it allows access to public and private container registries, the ability to push/pull images, and configure cloud integrations.

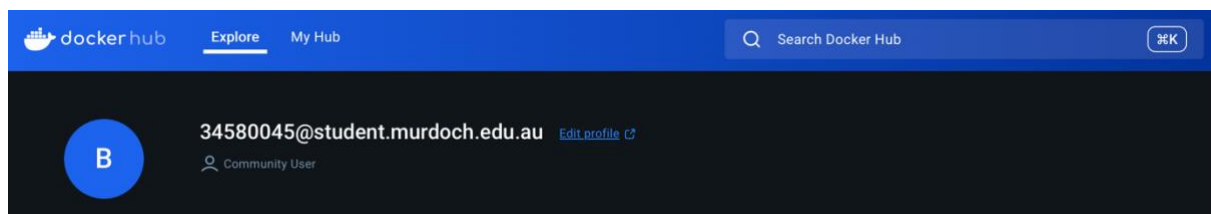


Figure 37: Docker Hub profile confirming successful sign-in using student credentials. This allows access to Docker image management, pull/push functionality, and future CI/CD integration.

4. Creating the First Container from My Project

After logging in, I initiated the creation of my first Docker container for this project using the docker-compose.yml file. This containerised my Ctrl + Shift + Game website by bundling all HTML and CSS content with the Apache server.

Using Docker Desktop and docker compose up, I successfully launched the container. This allowed me to test the application locally and ensure that all necessary dependencies were packaged correctly.

The container named ctrl-shift-apache was created and is now visible in the **Docker Desktop dashboard**, confirming that Docker is working correctly and is running my website.

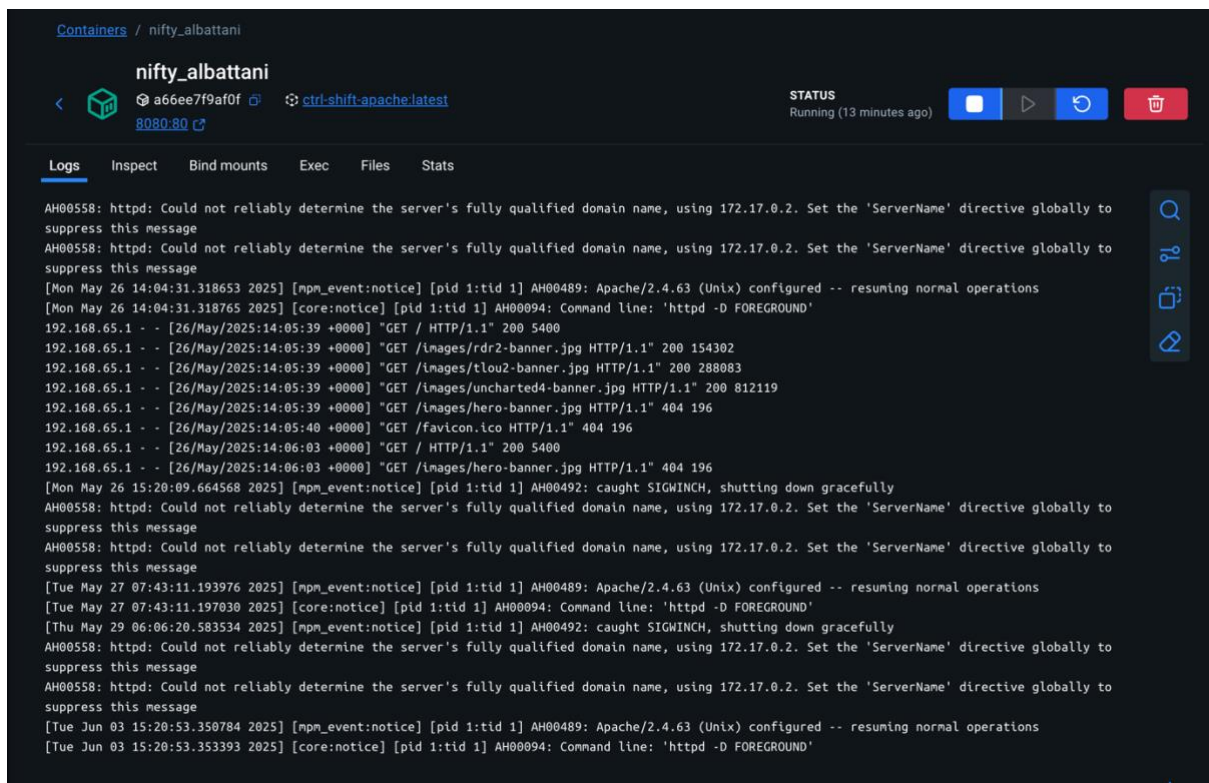


Figure 38 – Docker Desktop dashboard showing the ctrl-shift-apache container successfully running with port mapping and system resource usage.

5. Verifying the Installation

Once logged in, I verified that Docker was running properly. The whale icon appeared in the status bar, and the **Docker Dashboard** showed that the system was ready. This confirmed I could proceed to build and run containers for my application.

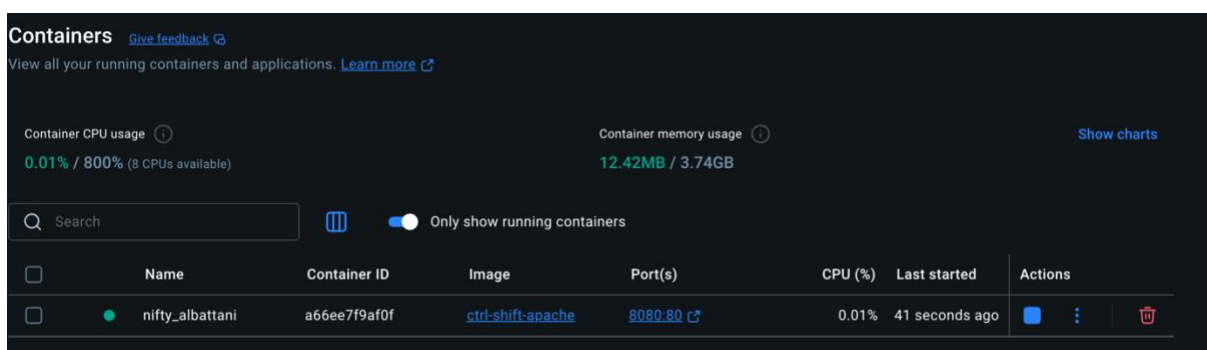


Figure 39: Detailed container overview in Docker Dashboard showing the status, image used, container name, mapped port (8080:80), and runtime statistics. This verifies successful container execution.

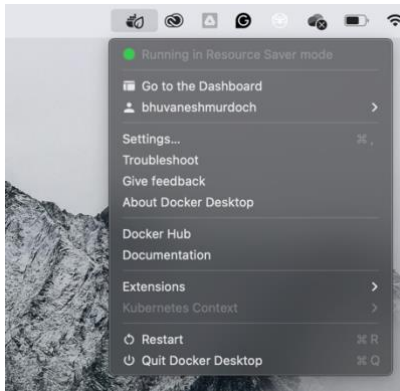


Figure 40: Whale icon in Mac status bar confirming Docker is running.

This initial setup lays the foundation for all containerised services used in my project and is a critical step before configuring the Dockerfile and docker-compose.yml in the following sections.

6.2 Docker Compose Configuration

A docker-compose.yml file was created to define the container settings. The container was named ctrl-shift-apache and mapped port 8080 of the host machine to port 80 of the container. The volume ./usr/local/apache2/htdocs/ was mounted so that the HTML and CSS files would serve through Apache inside the container.

```
docker-compose.yml
1  version: '3.8'
2
3  services:
4    ctrlshiftgame:
5      build: .
6      container_name: ctrl-shift-apache
7      ports:
8        - "8080:80"
9      volumes:
10       - ./usr/local/apache2/htdocs/
11      restart: unless-stopped
12
```

Figure 41: Docker Compose file configuring Apache container with port mapping and volume mount.

6.3 Docker Deployment Scripts

To further streamline the deployment of my website, I utilised scripting techniques within Docker. This section showcases the scripts used to build, run, and containerise the Apache server serving the Ctrl + Shift + Game website.

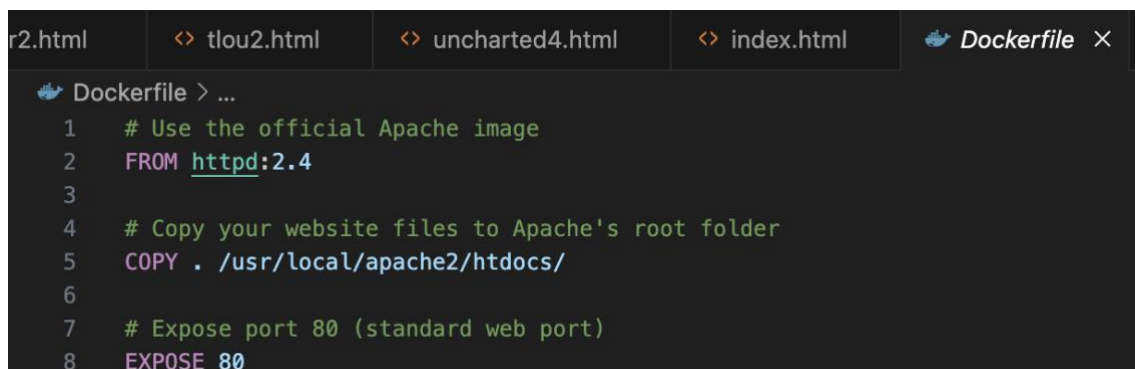
While this step was not required for the unit, including it demonstrates additional initiative and industry-relevant deployment skills such as scripting and containerization workflows.

6.3.1 Dockerfile Script

The Dockerfile defines the image and instructions required to build the container. It uses the official Apache image and copies all website files into the Apache document root (htdocs/). This script was placed in the project's root directory and used by Docker Compose during the container build process.

Script Purpose:

- Use httpd:2.4 image
- Copy website files
- Expose port 80

A screenshot of a code editor showing a Dockerfile. The editor has a dark theme and a tab bar at the top with files named 'r2.html', 'tlou2.html', 'uncharted4.html', 'index.html', and 'Dockerfile'. The Dockerfile content is as follows:

```
1  # Use the official Apache image
2  FROM httpd:2.4
3
4  # Copy your website files to Apache's root folder
5  COPY . /usr/local/apache2/htdocs/
6
7  # Expose port 80 (standard web port)
8  EXPOSE 80
```

Figure 42: Dockerfile opened in VS Code defining Apache image and file copy instructions

6.3.2 Automation Script (setup.sh)

To make deployment faster, I also created a basic bash script named setup.sh. This script automates the two main Docker commands:

```
docker build -t ctrlshiftgame .
docker-compose up -d
```

Script Purpose:

- Builds the Docker image
- Starts the container in detached mode using Docker Compose

```
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % docker ps
```

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
a66ee7f9af0f	ctrl-shift-apache	"httpd-foreground"	8 days ago	Up 54 minutes	0.0.0.0:8080->80/tcp

```
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % docker stop ctrl-shift-apache

ctrl-shift-apache
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % docker rm ctrl-shift-apache

ctrl-shift-apache
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % ./setup.sh

[+] Building 1.2s (7/7) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 219B                             0.0s
=> [internal] load metadata for docker.io/library/httpd:2.4     1.1s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 11.84kB                               0.0s
=> [1/2] FROM docker.io/library/httpd:2.4@sha256:09cb4b94edaaa796522c545328b62e9a0 0.0s
=> => resolve docker.io/library/httpd:2.4@sha256:09cb4b94edaaa796522c545328b62e9a0 0.0s
=> CACHED [2/2] COPY . /usr/local/apache2/htdocs/              0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:16aa9176050813f47ba15b4bcfacca68deb6ad3a6b1b39e542 0.0s
=> => exporting config sha256:e6ca5d03e854e394bac793d3e9397d76afc8b2f88e877bdbbc424 0.0s
=> => exporting attestation manifest sha256:e27549be58c3309f5f0e7f902ad3b263ecc002 0.0s
=> => exporting manifest list sha256:8e1008949de53db839dd2799bccfa86def04dad15f0cc 0.0s
=> => naming to docker.io/library/ctrlshiftgame:latest         0.0s
=> => unpacking to docker.io/library/ctrlshiftgame:latest      0.0s
WARN[0000] /Users/bhuvanesh/ict171-assignment2/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 0/1
  Container ctrl-shift-apache Starting                          0.2s
Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint ctrl-shift-apache (a7ebbafe2044f8dbd0aceb2099e5b08aa49f934bc91c65b894b2b554945716397): Bind for 0.0.0.0:8080 failed: port is already allocated
```

Figure 43: Initial Docker container startup failed due to port conflict on 8080. This demonstrates the importance of stopping existing containers before running a new one.

```

807200000742000774072007771 Bind for 0.0.0.0:80 failed: port is already allocated
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % docker stop nifty_albattani

nifty_albattani
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % docker rm nifty_albattani

nifty_albattani
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 % ./setup.sh

[+] Building 2.3s (8/8) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 219B                             0.0s
=> [internal] load metadata for docker.io/library/httpd:2.4     2.2s
=> [auth] library/httpd:pull token for registry-1.docker.io     0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 11.84kB                              0.0s
=> [1/2] FROM docker.io/library/httpd:2.4@sha256:09cb4b94edaaa796522c545328b62e9a0db 0.0s
=> => resolve docker.io/library/httpd:2.4@sha256:09cb4b94edaaa796522c545328b62e9a0db 0.0s
=> CACHED [2/2] COPY . /usr/local/apache2/htdocs/              0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:16aa9176050813f47ba15b4bcfacca68deb6ad3a6b1b39e542ec 0.0s
=> => exporting config sha256:e6ca5d03e854e394bac793d3e9397d76afc8b2f88e877bdbc4247f 0.0s
=> => exporting attestation manifest sha256:0055330e3207d50730c8cd71507be97a3c465283 0.0s
=> => exporting manifest list sha256:6725983f9d2489c0dbf0a942e0a07e0329bf26fd537926f 0.0s
=> => naming to docker.io/library/ctrlshiftgame:latest         0.0s
=> => unpacking to docker.io/library/ctrlshiftgame:latest      0.0s
WARN[0000] /Users/bhuvanesh/ict171-assignment2/docker-compose.yml: the attribute `version` i
s obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/1
✓ Container ctrl-shift-apache Started                          0.1s
(base) bhuvanesh@MacBook-Air-71 ict171-assignment2 %

```

Figure 44: Successful execution of `setup.sh` script after resolving port conflict, confirming the containerized Apache server was launched successfully.

2.html
< > tlou2.html
< > uncharted4.html
< > index.html
Dockerfile
\$ setup.sh U X

```

$ setup.sh
1  #!/bin/bash
2  docker build -t ctrlshiftgame .
3  docker-compose up -d

```

Figure 45: Final version of the `setup.sh` script used to automate Docker image build and container launch using Docker Compose. This enhances deployment speed and reduces manual errors.

7.0 Final Testing and Verification

7.1 Web Application Access and Security

To confirm the success of my web server deployment, I conducted thorough end-to-end testing to ensure all components function as intended across devices and browsers.

7.1.1 Website Accessibility via DNS

The website is publicly accessible using the DNS address <https://bhuvi.xyz>, which maps to the EC2 instance through GoDaddy's DNS A record configuration. The successful propagation of the DNS settings was verified by loading the site on multiple browsers and devices, and the content was consistently accessible.

The updated homepage, built with custom HTML and CSS, now reflects the final version of my blog, replacing the placeholder landing page seen earlier in Section 3.2.

- Refer to **Figure 47 – Screenshot of the final Ctrl + Shift + Game homepage deployed at <https://bhuvi.xyz> via Dockerized Apache server.**

7.1.2 SSL/TLS Functionality

The SSL certificate issued via **Let's Encrypt Certbot** is valid and active. The green padlock icon appears in the address bar, indicating the connection is secure. This ensures all data between the client and server is encrypted, fulfilling best practice for web hosting and required rubric criteria.

- Refer to **Figure 20 – Browser showing secure connection and valid SSL certificate for the domain.**

7.1.3 Docker Container Persistence

Using Docker Compose with the restart: unless-stopped directive ensures that the container automatically restarts after a system reboot or crash. I tested this by restarting Docker Desktop and verifying via:

```

Last login: Wed Jun  4 00:13:43 on ttys003
[(base) bhuvanesh@MacBook-Air-71 ~ % docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS                NAMES
fb0f1c0c5722   ict171-assignment2-ctrlshiftgame    "httpd-foreground"     38 minute
s ago         Up 34 minutes       0.0.0.0:8080->80/tcp    ctrl-shift-apache

```

Figure 46: Active Docker container (ctrl-shift-apache) running after reboot using restart policy.

The container (ctrl-shift-apache) remained active after reboot, confirming deployment resilience.

- Refer to **Figure 46**: Active Docker container (ctrl-shift-apache) running after reboot using restart policy.

7.1.4 Page Routing and Functional Testing

I manually navigated through the main site and each game-specific page to validate content rendering, routing, and media loading:

- / → Main homepage with navigation buttons.
- /rdr2.html → Red Dead Redemption 2 page with character profiles.
- /tlou2.html → The Last of Us Part II page.
- /uncharted4.html → Uncharted 4: A Thief's End page.

All pages displayed correctly, with images, text, and layout loading as designed.

- Refer to **Figure 47 – 51** – Screenshots of individual game pages confirming successful routing.

7.2 Final Live Website Deployment Screenshot

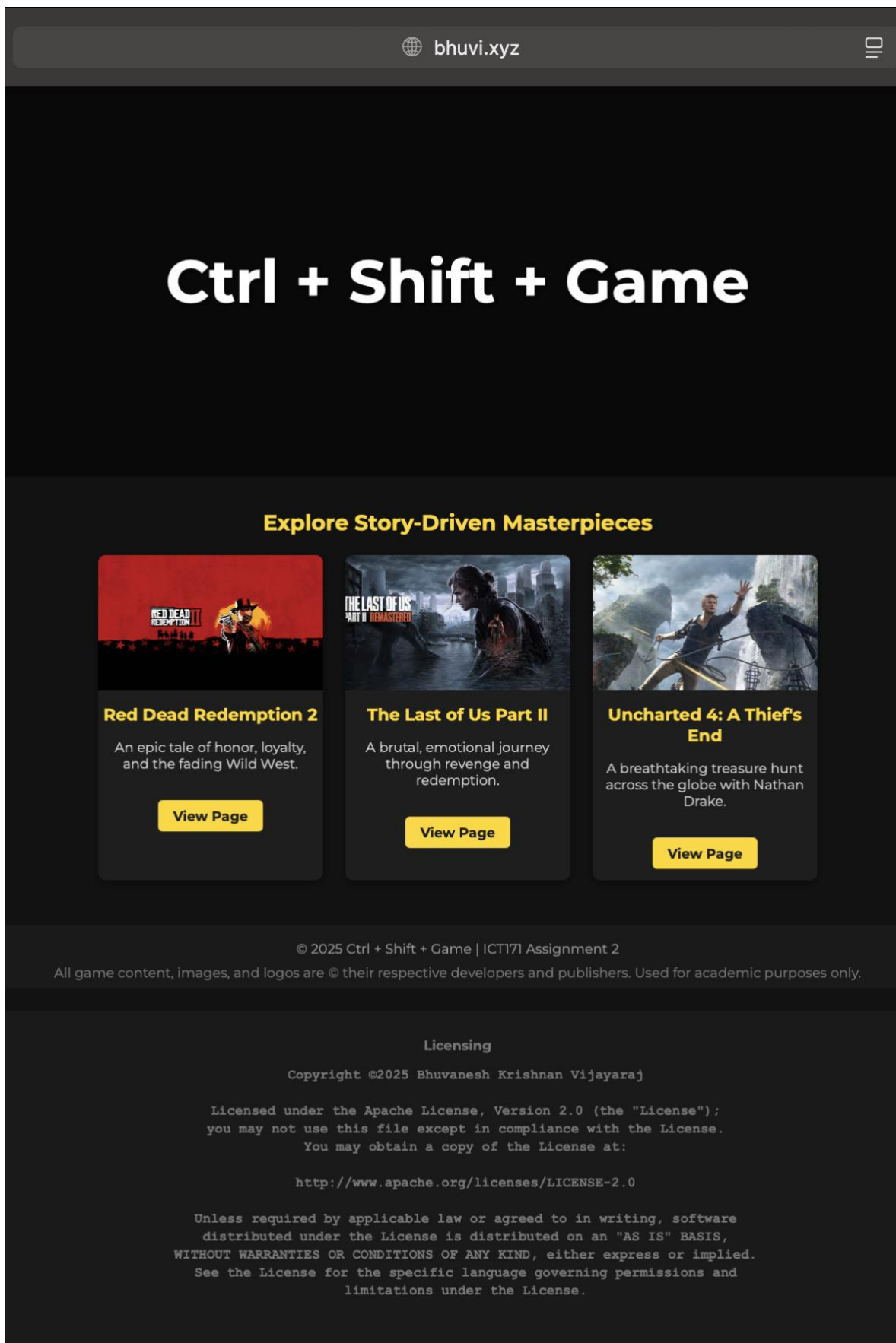


Figure 47: www.bhuvi.xyz main homepage with navigation buttons.

ROCKSTAR GAMES PRESENTS RED DEAD REDEMPTION II



Story Summary

America, 1899. The era of the Wild West is drawing to a close as lawmen hunt down the last remaining outlaw gangs. Those who will not surrender or succumb are killed. After a robbery goes badly wrong in the western town of Blackwater, Arthur Morgan and the Van der Linde gang are forced to flee. With federal agents and the best bounty hunters in the nation massing on their heels, the gang must rob, steal and fight their way across the rugged heartland of America in order to survive. As deepening internal divisions threaten to tear the gang apart, Arthur must make a choice between his own ideals and loyalty to the gang who raised him.

Main Characters



Arthur Morgan

Arthur is the game's main protagonist, a loyal yet conflicted senior enforcer of the Van der Linde gang. He's



Dutch van der Linde

Dutch is the charismatic, idealistic, and increasingly unhinged leader of the Van der Linde gang. He views himself



John Marston

John is another key member of the gang and the protagonist of the original Red Dead Redemption. He's a capable

Figure 48: www.bhuvi.xyz Red Dead Redemption 2 page

THE LAST OF US PART II REMASTERED

Story Summary

Five years after their dangerous journey across the post-pandemic United States, Ellie and Joel have found a semblance of peace in Jackson, Wyoming. However, when a violent event disrupts that peace, Ellie embarks on a relentless quest for vengeance and justice. Her pursuit takes her through a hostile world, forcing her to confront the devastating physical and emotional consequences of her actions. The Last of Us Part II is a harrowing exploration of the cycles of violence, the complexities of trauma, and the blurred lines between hero and villain in a world where survival often demands brutal choices.

Main Characters



Ellie

Ellie is the main protagonist, hardened by loss and driven by an all-consuming need for vengeance.



Abby Anderson

Abby's story intertwines with Ellie's, offering a contrasting perspective. She is a strong and conflicted



Joel Miller

Joel is Ellie's protective father figure. His past actions and complex choices at the end of the first game

Figure 49: www.bhuvi.xyz The Last of Us Part II page.



Story Summary

Several years after his last adventure, retired fortune hunter Nathan Drake is forced back into the world of thieves when his presumed-dead brother, Sam, resurfaces. Together, they embark on a globetrotting hunt for Captain Avery's long-lost pirate treasure, testing Drake's limits and what he's willing to sacrifice.

Main Characters



Nathan Drake

The charismatic and resilient treasure hunter, now contemplating retirement.



Samuel "Sam" Drake

Nathan's older brother, whose reappearance kickstarts the adventure.



Elena Fisher

Nathan's wife, an investigative journalist who often gets pulled into his escapades.

[Back to Home](#)

7.3 Troubleshooting Tips

Below are common issues encountered during the deployment of the **Ctrl + Shift + Game** web server and how they were resolved. These practical notes can help replicate and debug the server setup effectively.

7.3.1 Apache Not Reflecting Updated Files

Issue: Images or changes to .html or .css were not updating in the browser, even after using scp to upload new files.

Cause: The Docker volume might be persisting old content or the browser cached the image.

Fix:

- First, ensure you **run this full rebuild sequence:**

```
docker-compose down --volumes --remove-orphans
docker-compose build --no-cache
docker-compose up -d
```

7.3.2 Docker Port Conflict

Issue: Running docker-compose up -d gave a “port already in use” error.

Cause: An older container using port 8080 was already running.

Fix:

- Stop all containers:

```
docker stop $(docker ps -q)
```

- Then restart cleanly:

```
docker-compose up -d
```


7.3.3 SSL Certificate Not Valid After Setup

Issue: certbot completed, but <https://bhuvi.xyz> still showed “Not Secure.”

Fix:

- Restart Apache manually:

```
sudo systemctl restart apache2
```

- Ensure port 443 is allowed in the EC2 security group inbound rules.
- Check with a DNS propagation tool like <https://dnschecker.org> and try again in an incognito window.

7.3.4 Website Doesn't Load After EC2 Reboot

Issue: After rebooting EC2 or Docker Desktop, the site was no longer accessible.

Fix:

- Docker Compose was set with:

```
restart: unless-stopped
```

This ensures the container auto-restarts. You can confirm with:

```
docker ps
```

If it's not running, restart manually:

```
docker-compose up -d
```

7.3.5 Files Not Loading Inside Container

Issue: Uploaded files were not visible in the site after scp.

Cause: Files were uploaded **to the host system**, but not properly reflected in the container's `/usr/local/apache2/htdocs/`.

Fix:

- Make sure your `docker-compose.yml` has the correct volume mount:

```
volumes:  
- ./ : /usr/local/apache2/htdocs/
```

- Rebuild and relaunch the container.

7.3.6 Certbot Not Installing SSL

Issue: Running `sudo certbot --apache` failed midway.

Fix:

- Make sure Apache is running:

```
sudo systemctl start apache2
```

- Ensure **port 80 and 443** are open in the AWS EC2 security group.
- Retry the command or manually specify domains:

```
sudo certbot --apache -d bhuvi.xyz -d www.bhuvi.xyz
```

8.0 Licensing

This project, *Ctrl + Shift + Game*, was developed as part of Murdoch University's ICT171 Assignment 2 and is strictly intended for **educational use only**.

- The web server is powered by **Apache HTTP Server**, used via the official Docker image (`httpd:2.4`), which is licensed under the **Apache License 2.0**.
See: <https://www.apache.org/licenses/LICENSE-2.0>
- All game titles, images, character names, and logos used (e.g., *Red Dead Redemption 2*, *The Last of Us Part II*, *Uncharted 4*) remain the intellectual property of their respective developers and publishers (e.g.,

Rockstar Games, Naughty Dog, Sony Interactive Entertainment).

- All content is presented under **fair use** for academic, non-commercial purposes only. No assets are redistributed or monetised.
- This server and codebase are privately hosted and not intended for public deployment, monetisation, or commercial redistribution.

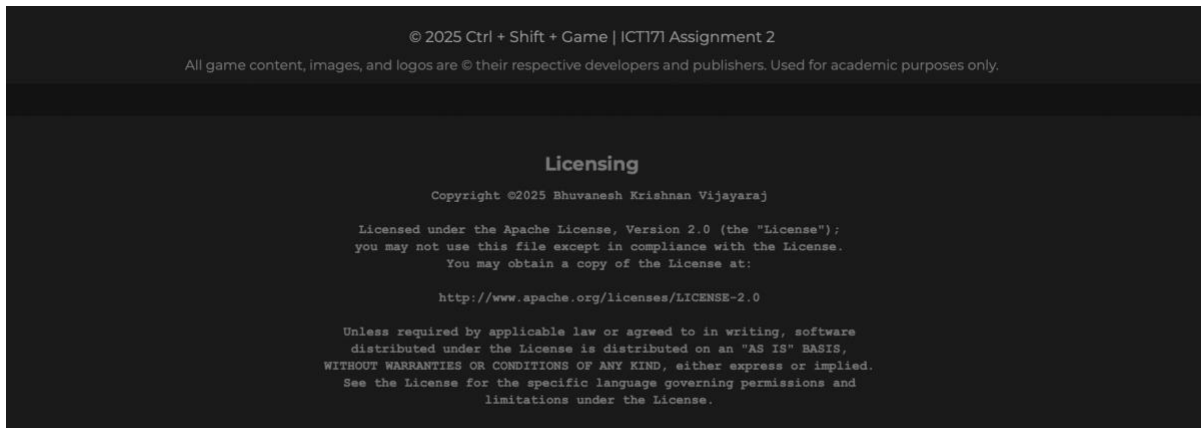


Figure S1: www.bhuvi.cxz End of homepage licensing page