/anu\_hospitality\_project/

│

├── index.php # Landing page (Home + About + Services + Job Roles + Contact)

├── services.php # Detailed Services page

├── job\_roles.php # Detailed Job Roles page

├── candidate\_apply.php # Candidate job application form

├── company\_join.php # Company partner form

├── policy\_privacy.php # Privacy Policy

├── policy\_terms.php # Terms & Conditions

├── policy\_refund.php # Refund Policy

├── policy\_cookies.php # Cookies Policy

├── policy\_other.php # Other Policies (optional)

│

├── admin/ # Admin Panel & Backend

│ ├── login.php # Admin login

│ ├── forgot\_password.php # Admin password reset

│ ├── dashboard.php # Admin Dashboard (Main Panel)

│ ├── candidates.php # Candidate data table with search & pagination

│ ├── companies.php # Companies data table

│ ├── queries.php # Contact queries table

│ ├── logout.php # Logout script

│ └── assets/ # Admin-specific CSS/JS/images if needed

│

├── includes/ # Reusable components & secure DB

│ ├── header.php # Header included in all pages

│ ├── footer.php # Footer included in all pages

│ └── db\_connect.php # Database connection (DO NOT push real credentials)

│

├── config/ # Secure configuration & secrets (future-proof)

│ └── config.php # Example: DB credentials, API keys, environment variables

│

├── assets/ # Global assets for the website

│ ├── css/

│ │ └── style.css # Main CSS

│ ├── js/

│ │ └── script.js # Main JS

│ └── images/

│ └── logo.png # Logo & other images

│

├── .htaccess # Clean URLs (.php hidden), redirects, security rules

├── .gitignore # Ignore sensitive files (config/, db\_connect.php)

├── README.md # Project description & setup instructions

**.htaccess for Your Project**

# ========================================

# Enable Rewrite Engine for Clean URLs

# ========================================

RewriteEngine On

# ----------------------------------------

# Remove .php extension from URLs

# Example: /about loads /about.php

# ----------------------------------------

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_FILENAME}\.php -f

RewriteRule ^(.\*)$ $1.php [L]

# ----------------------------------------

# Redirect index.php to root domain

# ----------------------------------------

RewriteCond %{THE\_REQUEST} /index\.php

RewriteRule ^index\.php$ / [R=301,L]

# ========================================

# Security Settings

# ========================================

# Prevent directory browsing

Options -Indexes

# Block access to sensitive files

<FilesMatch "(\.env|config\.php|db\_connect\.php)">

Order allow,deny

Deny from all

</FilesMatch>

# Protect .htaccess itself

<Files .htaccess>

Order allow,deny

Deny from all

</Files>

# ========================================

# Optional: Redirect www to non-www (choose one)

# ========================================

# RewriteCond %{HTTP\_HOST} ^www\.(.\*)$ [NC]

# RewriteRule ^(.\*)$ https://%1/$1 [R=301,L]

# ========================================

# Optional: Force HTTPS (if SSL is installed)

# ========================================

# RewriteCond %{HTTPS} off

# RewriteRule ^(.\*)$ https://%{HTTP\_HOST}/$1 [R=301,L]

**✅ How It Works**

1. **Clean URLs**
   * https://yourdomain.com/about → loads about.php
   * https://yourdomain.com/job\_roles → loads job\_roles.php
2. **Redirect index.php**
   * https://yourdomain.com/index.php → automatically redirects to https://yourdomain.com/
3. **Security**
   * Blocks access to sensitive files (.env, config.php, db\_connect.php)
   * Prevents directory listing (Options -Indexes)
   * Protects .htaccess itself
4. **Optional**
   * Redirect www to non-www for professional URLs
   * Force HTTPS if SSL is installed

**💡 Tips**

* Place .htaccess in the **root folder** (/anu\_hospitality\_project/)
* Test each page after uploading to make sure clean URLs work
* Make backups before changing rules

About services section

**1️⃣ Staff & Workforce Solutions**

* **Professional Hospitality Staff**  
  Trained waiters, chefs, housekeeping, front desk staff.
* **Event Staff Management**  
  Staff for weddings, conferences, corporate events.
* **Temporary & Permanent Staffing**  
  Flexible solutions for hotels, resorts, restaurants.

**2️⃣ Catering & Food Services**

* **In-House Catering**  
  Full-service catering for events and corporate clients.
* **Custom Menus**  
  Tailored menus for weddings, parties, or corporate events.
* **Buffet & Banquet Management**  
  Food preparation, serving, and cleanup.

**3️⃣ Training & Consultancy**

* **Hospitality Staff Training**  
  Grooming, etiquette, service standards, and hospitality skills.
* **Consultancy Services**  
  Hotel or restaurant workflow management, operational guidance.
* **Customer Service Excellence Programs**  
  Improve guest experience and feedback management.

**4️⃣ Facility & Event Management**

* **Venue Setup & Decoration**  
  Organizing corporate events, weddings, and parties.
* **Logistics & Coordination**  
  Coordinating staff, equipment, and schedules for events.
* **Special Requests Handling**  
  Luxury or customized client experiences.

**5️⃣ Premium Services**

* **VIP Concierge Services**  
  Personal staff for executives or high-profile guests.
* **Luxury Event Staffing**  
  Trained staff for high-end weddings, corporate parties.
* **On-Demand Hospitality Services**  
  Temporary staff deployment for emergencies or peak seasons.

### ****6th Event & Banquet Management Staff****

* Provide trained staff for **weddings, parties, corporate events**, and other large gatherings.
* Includes **waiters, coordinators, bartenders, and cleaning staff**.
* Ensures **smooth operations** for any type of event

Excellent question — and your Git output actually tells us **exactly what’s happening**. Let’s break it down clearly and simply 👇

**🧩 Step-by-step Meaning of Your Output**

**1️⃣**

git init

Reinitialized existing Git repository in D:/Kishan\_Solanki\_Projects/Clients\_Projects/Job\_Portal\_Website/.git/

✅ This means you already had a .git folder inside your project (it’s already a Git repo).  
Git just **reinitialized** it — nothing harmful, but it didn’t create a *new* repo; it reused the existing one.

**2️⃣**

git add .

git commit -m "add index"

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

nothing to commit, working tree clean

✅ This means:

* You **already committed** your latest changes earlier.
* The “add index” commit didn’t add anything new (since your working directory is clean).
* You are **ahead of the remote branch by 1 commit**, meaning your local repo has one extra commit that hasn’t yet been pushed.

**3️⃣**

git push

Enumerating objects: 66, done.

...

error: RPC failed; curl 55 Recv failure: Connection was reset

...

fatal: the remote end hung up unexpectedly

Everything up-to-date

❌ This part is the **real issue** — Git tried to send your commit to the remote repository but **the network connection was interrupted or the remote server rejected it**.

**⚙️ Why This Happens (in Simple Words)**

This happens when Git tries to send a **large amount of data** to your remote (GitHub / GitLab / Bitbucket), and during that process:

* The connection drops or resets (unstable network).
* Or the **remote server times out** because the push is too big (for example, >100 MB file).
* Or Git’s **HTTP buffer size** is too small for the large data.

The key clue here is this line:

Writing objects: 100% (51/51), 127.72 MiB

So you’re pushing **~128 MB of data**, which is **very large** for a single Git push.

**🛠️ How to Fix It**

Here’s what you can do step-by-step:

**✅ Step 1 — Increase Git’s buffer size**

Run this once:

git config --global http.postBuffer 524288000

That sets the buffer to 500 MB (default is much smaller).

**✅ Step 2 — Disable compression (optional but helps)**

git config --global core.compression 0

**✅ Step 3 — If you have large files, use Git LFS**

Install and use **Git LFS (Large File Storage)** for files like images, videos, or zips:

git lfs install

git lfs track "\*.jpg"

git lfs track "\*.png"

git lfs track "\*.zip"

git add .gitattributes

git add .

git commit -m "track large files using git lfs"

git push origin main

**✅ Step 4 — Try pushing again**

Now try:

git push origin main

If it still fails, try again after ensuring your internet is stable or use a wired connection.

**✅ Step 5 — (If still fails)**

Split the commit:

git reset HEAD~1

git add some\_files

git commit -m "partial commit 1"

git push

git add remaining\_files

git commit -m "partial commit 2"

git push

**✅ Step 6 — (Optional: Check what’s big)**

You can identify large files by running:

git lfs migrate info

or

git rev-list --objects --all | sort -k 2 > allfiles.txt

Then open allfiles.txt to see what’s large.

**💡 Summary**

| **Problem** | **Cause** | **Fix** |
| --- | --- | --- |
| RPC failed / connection reset | Large push or slow internet | Increase buffer size |
| Remote end hung up | Timeout or large file | Use Git LFS or split commits |
| “Everything up-to-date” after error | Misleading message | Push likely failed, re-run after fix |

If you want, you can share your **list of recently added files (like images, videos, zips, etc.)**, and I’ll tell you which ones are likely causing the push failure.  
Would you like me to show you the exact command to list files above, say, **50 MB** in your repo?

Need to change in contact\_submit.php and in candidate\_submit.php

: $mail->Host = 'smtp.yourdomain.com'; // your SMTP host

$mail->Username = 'your\_email@domain.com'; // your email

$mail->Password = 'your\_email\_password'; // email password

$mail->setFrom('your\_email@domain.com', 'ANU Hospitality Staff');

$mail->addAddress('admin@anuhospitality.com', 'Admin'); // admin email

**File permissions**

**is necessary** for folders where your PHP code needs to **write files**, like the uploads/resumes/ folder for candidate resumes. Without correct permissions, PHP won’t be able to save the uploaded files, and candidates will get errors.

Here’s how to handle it on **Hostinger**:

**1️⃣ Using Hostinger File Manager**

1. Log in to your **Hostinger hPanel**.
2. Go to **File Manager**.
3. Navigate to your project folder → uploads/resumes/.
4. Right-click on the folder → **Change Permissions / Permissions**.
5. Set it to **755**:
   * Owner → Read, Write, Execute ✅
   * Group → Read, Execute ✅
   * Public/Others → Read, Execute ✅
6. Save changes.

✅ This allows PHP to save files safely without giving write access to everyone.

**2️⃣ Using FTP (FileZilla, WinSCP, etc.)**

1. Connect to your hosting via FTP.
2. Navigate to uploads/resumes/.
3. Right-click → **File permissions** or **CHMOD**.
4. Enter **755**.
5. Apply recursively if needed (for subfolders).

**🔹 When 777 is needed**

* Only if uploads **still fail**, sometimes servers require full write access.
* Example: 777 = anyone can read/write/execute (less secure).

So yes, you **must set folder permissions** at least for the upload folder. Other files/folders (like PHP code) don’t need it—keep them at 644 or 755 for security.

**1️⃣ Root Project Folder**

* **Path:** /public\_html/ or wherever your site is hosted.
* **Permission:** 755
* **Reason:** PHP needs to read/execute files; no write needed by public.

**2️⃣ PHP Files / Includes**

* **Path:** includes/, PHPMailer/, any .php files.
* **Permission:** 644
* **Reason:** Owner can read/write, server can read; public cannot write.

**3️⃣ Uploads Folder (for resumes, images, etc.)**

* **Path:** uploads/ and uploads/resumes/
* **Permission:** 755 (default) ✅
  + If uploads **fail**, temporarily use 775 or 777.
* **Reason:** PHP needs write access to save files. Public does **not** need write access.

**4️⃣ CSS, JS, Images**

* **Path:** assets/css/, assets/js/, assets/images/
* **Permission:** 644
* **Reason:** Only server needs to read and serve files; no writing required.

**5️⃣ Configuration Files**

* **Path:** includes/db\_connect.php, includes/env.php
* **Permission:** 600 or 644
* **Reason:** Contains sensitive info (DB credentials, email password). Only the owner should ideally write; server reads.

**How to Set on Hostinger**

**Using File Manager**

1. Navigate to folder/file → Right-click → **Change Permissions**
2. Set numbers as above (755, 644, 600, etc.)
3. For uploads folder: make **recursive** if it has subfolders.

**Using FTP / SSH**

* **Command for folder:**

chmod 755 folder\_name

* **Command for files:**

chmod 644 file\_name

* **Recursive for uploads folder:**

chmod -R 755 uploads/

✅ With this setup:

* Your **site works** (uploads succeed).
* Your **files are secure** from public write access.
* Sensitive info like DB credentials and SMTP password are protected.