Web Application Programming

Week 13. Web Application Deployment and Maintenance

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Web Application Deployment and Maintenance(
   i. Hosting options and
   ii. deployment strategies,
   iii. Debugging and testing web applications
   iv. Best practices for maintaining and updating web applications)
   v. Register a with a hosting company, register a domain and host our files

Web Application Deployment and Maintenance

Web Application Deployment

Web application deployment involves making your web application accessible and operational on a server or hosting environment.

Here's a step-by-step guide for deploying a web application

1. **Code Version Control:**
   i. Use a version control system like Git to manage your codebase.
   ii. Ensure your code is in a stable state before deployment.
2. **Prepare for Deployment:**
   i. Check that your application's dependencies are up-to-date.
   ii. Update configuration files for the production environment.
   iii. Remove unnecessary files and perform any pre-deployment tasks.

Web Application Deployment

Step-by-step guide for deploying a web application

3. Environment Setup:

i. Choose a hosting provider or set up your own server.

ii. Configure the server with the necessary software stack (web server, database server, etc.).

4. Database Migration:

i. If your application involves a database, perform any necessary database migrations to update the schema.

ii. Ensure database connections and configurations are set for the production environment.

Web Application Deployment

Step-by-step guide for deploying a web application

5. Security Measures:

   i. Set up SSL certificates to enable HTTPS.

   ii. Configure firewalls and security groups.

   iii. Ensure that sensitive information like API keys and database credentials are securely stored.

6. Build and Compile:

   i. If your application uses a compiled language, build the code for the production environment.

   ii. Compile assets like stylesheets and JavaScript files.

Web Application Deployment

Step-by-step guide for deploying a web application

7. Testing:

   i. Perform thorough testing on a staging environment that closely mimics the production environment.

   ii. Test functionality, performance, and security.

8. Continuous Integration/Continuous Deployment (CI/CD):

   i. If you have a CI/CD pipeline, trigger the deployment process.

   ii. This might involve automated testing and deployment to different environments.

Web Application Deployment

Step-by-step guide for deploying a web application

9. Deploy the Application:
    i. Upload your application code and assets to the production server.
    ii. Ensure all configurations are correctly set for the production environment.
10. Web Server Configuration:
    i. Configure the web server (e.g., Apache, Nginx) to serve your application.
    ii. Set up virtual hosts or server blocks.
11. Database Connection:
    i. Verify that the production database is connected and accessible from the deployed application.

Web Application Deployment

Step-by-step guide for deploying a web application

12. Monitoring and Logging:
    i. Set up monitoring tools to track the application's performance and health.
    ii. Configure logs to capture relevant information for debugging.
13. Final Checks:
    i. Perform final checks on the production environment.
    ii. Check that all services (web server, database server, etc.) are running.
14. Rollback Plan:
    i. Have a rollback plan in case issues arise after deployment.
    ii. This might involve reverting to a previous code version or database backup.

Web Application Deployment

Step-by-step guide for deploying a web application

15. Update DNS Records:
    i. If applicable, update DNS records to point to the new production server.
16. Post-Deployment Testing:
    i. Perform additional testing on the live environment to ensure everything works as expected.
17. Notify Stakeholders:
    i. Communicate with stakeholders about the successful deployment.
    ii. Inform them of any expected downtime during the deployment process.

Web Application Deployment

Step-by-step guide for deploying a web application

18. Documentation:

i. Update deployment documentation with any changes made during the deployment process.

ii. Document any issues encountered and their resolutions.

By following these steps, you can ensure a smooth and controlled deployment of your web application. Regularly review and update your deployment process as your application evolves and requirements change.

Web Application Deployment Strategies

Web Application Deployment Strategies

Web application deployment strategies involve the process of releasing a new version of a web application or updating an existing one. The goal is to make changes to the production environment while minimizing downtime, ensuring a smooth transition, and reducing the risk of errors. Here are some **common web application deployment strategies:**

1. **Blue-Green Deployment:**

- **Overview:** Involves running two identical production environments, one (let's say, Blue) currently in use, and the other (Green) inactive.
- **Deployment Process:**
  i. Deploy the new version to the inactive environment (Green).
  ii. Switch the router or load balancer to direct traffic to the Green environment.
  iii. The Blue environment is now inactive and can be updated for the next deployment.

Web Application Deployment Strategies

Common web application deployment strategies:

2. Canary Deployment:

- **Overview:** Gradual rollout of the new version to a small subset of users before making it available to the entire user base.
- **Deployment Process:**
  i. Deploy the new version to a small percentage of servers or users.
  ii. Monitor performance, errors, and user feedback.
  iii. If successful, gradually increase the rollout to a larger audience.

3. Rolling Deployment:

- **Overview:** Updates are deployed incrementally across the server pool, avoiding downtime.
- **Deployment Process:**
  i. Deploy the new version to a subset of servers.
  ii. Verify that the deployment is successful and monitor for any issues.
  iii. Gradually roll out the update to additional servers until all are updated.

Web Application Deployment Strategies

Common web application deployment strategies:

4. Feature Toggles (Feature Flags):

- **Overview:** Allows you to toggle features on or off at runtime, enabling selective deployment of specific features.
- **Deployment Process:**
  - Deploy the new version with feature toggles in the "off" position.
  - Gradually enable the new features for testing and validation.

5. A/B Testing:

- **Overview:** Involves deploying multiple versions of a feature or page to different groups of users to compare their performance.
- **Deployment Process:**
  - Deploy multiple versions of the feature.
  - Direct different user groups to different versions.
  - Analyze user behavior and feedback to determine the preferred version.

Web Application Deployment Strategies

Common web application deployment strategies:

6. Shadow Deployment:

- **Overview:** Involves running the new version in parallel with the existing version, but without directing actual user traffic to it.
- **Deployment Process:**
    i. Deploy the new version alongside the existing one.
    ii. Route a small percentage of production traffic to the new version for monitoring and testing purposes.

7. Rollback Strategy:

- **Overview:** Having a well-defined plan for reverting to the previous version in case of issues.
- **Deployment Process:**
    i. If issues are detected, quickly roll back to the previous version.
    ii. Investigate and address the problems before attempting the deployment again.

Web Application Deployment Strategies

Common web application deployment strategies

8. Immutable Deployment:

- **Overview:** Involves deploying the new version as a completely new set of infrastructure instead of updating the existing one.
- **Deployment Process:**
    i. Launch a new set of servers with the updated version.
    ii. Switch the router or load balancer to direct traffic to the new set of servers.

9. Emergency Rollback:

- **Overview:** A rapid rollback in the event of critical issues affecting the user experience or application functionality.
- **Deployment Process:**
    i. Detect critical issues.
    ii. Immediately roll back to the previous version to minimize impact.

Web Application Deployment Strategies

Common web application deployment strategies

10. Zero-Downtime Deployment:

- **Best Practices:**

i. **Automation:** Use automated deployment tools and scripts to reduce the chance of human error.

ii. **Testing:** Perform thorough testing, including unit tests, integration tests, and user acceptance tests, before deploying.

iii. **Monitoring:** Implement robust monitoring to detect issues early and during deployment.

iv. **Communication:** Keep stakeholders informed about deployment schedules, and communicate any issues or changes.

The choice of deployment strategy depends on factors like the application architecture, infrastructure, team expertise, and the specific goals of the deployment. It's common to combine multiple strategies to address different aspects of the deployment process.

Web Application maintenance

Web application maintenance is an ongoing process that involves keeping your web application running smoothly, securely, and up to date.

Here's a guide for effective web application maintenance:

1. **Regular Updates:**

- **Codebase:**
  i. Continuously update and maintain your codebase.
  ii. Apply patches, bug fixes, and enhancements.
- **Dependencies:**
  i. Regularly update third-party libraries and dependencies to patch security vulnerabilities and benefit from new features.

Web Application maintenance

A guide for effective web application maintenance

4. Backup and Disaster Recovery:

i. Regularly back up your data, including databases, configurations, and assets.

ii. Test the restoration process to ensure data can be recovered in case of a disaster.

5. Performance Optimization:

i. Regularly analyze and optimize your application for performance.

ii. Optimize database queries, server response times, and frontend assets.

iii. Implement caching mechanisms.

6. Scaling:

i. Monitor traffic patterns and scale resources accordingly.

ii. Optimize infrastructure for cost-effectiveness.

Web Application maintenance

A guide for effective web application maintenance

7. User Feedback and Support:

i. Collect and analyze user feedback to improve user experience.

ii. Provide timely support for reported issues.

8. Regulatory Compliance:

i. Ensure compliance with relevant data protection and privacy regulations.

ii. Regularly review and update privacy policies.

9. Documentation Updates:

i. Keep all documentation up to date, including system architecture, deployment processes, and troubleshooting guides.

Web Application maintenance

A guide for effective web application maintenance

10. Database Maintenance:

i. Regularly optimize and clean up the database.

ii. Monitor and manage database indexes for efficient queries.

11. Content Updates:

i. Regularly update content, especially if your web application involves dynamic or frequently changing information.

12. Scalability Testing:

i. Periodically conduct scalability testing to ensure your application can handle increased loads.

Web Application maintenance

A guide for effective web application maintenance

13. Training and Knowledge Transfer:

i. Ensure that the development and operations team members are trained on the latest technologies and best practices.

ii. Document institutional knowledge and share it within the team.

14. Health Checks:

i. Implement automated health checks to proactively identify and address issues.

ii. Regularly audit logs and monitoring data.

15. Cost Optimization:

i. Review and optimize infrastructure costs regularly.

ii. Identify and eliminate any unused or unnecessary resources.

Web Application maintenance

A guide for effective web application maintenance

16. Technology Updates:

i. Stay informed about updates to your technology stack.

ii. Plan and execute updates to avoid using outdated or unsupported software.

17. Legal Compliance:

i. Regularly review and update terms of service and other legal documents to ensure compliance with changing regulations.

Web Application maintenance

A guide for effective web application maintenance

18. Review and Improve:

i. Periodically review your application architecture and overall performance.

ii. Identify areas for improvement and plan for future enhancements.

By consistently implementing these maintenance practices, you can ensure the long-term health and success of your web application. Regularly reassess and adapt your maintenance strategy based on evolving requirements and technological advancements.

Web App Hosting options

Web application hosting refers to the process of making a web application accessible and operational on a server or hosting environment.

This involves deploying the application, configuring the necessary infrastructure, and ensuring that it can be accessed by users over the internet.

The choice of hosting option depends on various factors such as the size and complexity of the application, anticipated traffic, performance requirements, and budget considerations.

Web App Hosting options

Popular hosting options

1. Shared Hosting:

i. Ideal for small websites and beginners.

ii. Multiple websites share the same server resources.

iii. Cost-effective but may have limitations on performance and customization.

2. Virtual Private Server (VPS):

i. Provides dedicated resources on a virtualized server.

ii. More control and flexibility compared to shared hosting.

iii. Suitable for medium-sized websites with moderate traffic.

3. Cloud Hosting:

i. Uses virtual servers from a cloud service provider (e.g. Azure, Google Cloud).

ii. Offers scalability, flexibility, and often a pay-as-you-go pricing model.

iii. Suitable for a wide range of applications, from small to large.

Web App Hosting options

Popular hosting options

4. **Dedicated Server Hosting:**
   i. Provides an entire physical server dedicated to your application.
   ii. Offers maximum control and performance.
   iii. Suitable for large, resource-intensive applications.

5. **Managed WordPress Hosting:**
   i. Specifically designed for WordPress websites.
   ii. Optimized for performance, security, and ease of use.
   iii. Managed services handle updates, backups, and security.

6. **Reseller Hosting:**
   i. Allows individuals or businesses to resell hosting services.
   ii. Ideal for web developers or agencies managing multiple client websites.

Web App Hosting options

Popular hosting options

8. **Colocation Hosting:**
   i. You own the server hardware, but it's housed in a data center.
   ii. You manage and maintain the server while leveraging the data center's infrastructure.
9. **Serverless Computing:**
   i. Runs applications without provisioning or managing servers.
   ii. Automatically scales based on demand.
   iii. Suitable for event-driven and microservices architectures.
10. **Content Delivery Network (CDN):**
   i. Distributes static content to servers worldwide for faster delivery.
   ii. Enhances website performance and reduces latency.
   iii. Often used in conjunction with other hosting options.

Web App Hosting options

Popular hosting options

11. **Edge Computing:**
    i. Moves computation closer to the data source (edge of the network).
    ii. Reduces latency and improves performance for geographically distributed applications.
12. **Clustered Hosting:**
    i. Involves using multiple servers to distribute the load.
    ii. Provides redundancy and ensures high availability.

Web App Hosting options

Popular hosting options

13. **Hybrid Hosting:**
    i. Combines on-premises infrastructure with cloud-based services.
    ii. Offers flexibility and scalability while retaining some level of control.

Consider factors such as traffic volume, scalability, security requirements, budget, and technical expertise when choosing a hosting option. It's common for larger applications to use a combination of these hosting options, leveraging each for its specific strengths within the overall architecture.

Debugging and Testing web Applications

Debugging and testing are critical phases in the development life cycle of web applications.

Here's an overview of key strategies and tools for debugging and testing web applications:

Debugging:

1. Browser Developer Tools:

Overview: Every modern browser provides developer tools that include elements inspection, network monitoring, console for logging, and JavaScript debugging.

Usage: Use breakpoints, step-through debugging, and console logging for JavaScript debugging.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

2. Logging:

Overview: Use **console.log** statements in your JavaScript code to output information to the browser console.

Usage: Insert logs strategically to trace the flow of your application and identify issues.

3. Debugger Statements:

Overview: Insert debugger; statements in your JavaScript code to pause execution and open the browser's developer tools.

Usage: Helps to step through code and inspect variables at specific points.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

4. Error Tracking Tools:

- Overview: Use error tracking services (e.g., Sentry, Rollbar) to monitor and log errors in the production environment.
- Usage: Receive alerts and detailed error reports to diagnose and fix issues.

5. Network Tab (Browser DevTools):

- Overview: Inspect network requests and responses to identify issues related to HTTP requests, status codes, and data payloads.
- Usage: Check for errors, slow-loading resources, and unexpected responses.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

5. Cross-Browser Testing:

- **Overview:** Use tools like BrowserStack or CrossBrowserTesting to test your application on different browsers and versions.
- **Usage:** Ensure compatibility and identify browser-specific issues.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

Testing:

1. Manual Testing:

- **Overview:** Test your application manually by interacting with it as an end-user.
- **Usage:** Identify UI/UX issues, functional bugs, and usability concerns.

2. Automated Testing:

- **Overview:** Write automated tests for your application using tools like Selenium (for end-to-end testing) or Cypress.
- **Usage:** Automate repetitive test scenarios, regression tests, and critical paths.

3. Load Testing:

- **Overview:** Use tools like Apache JMeter or Locust to simulate high traffic and analyze how your application handles load.
- **Usage:** Identify performance bottlenecks, server load issues, and optimize accordingly.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

Testing:

4. Security Testing:

- **Overview:** Conduct security audits and use tools like OWASP ZAP or Burp Suite for penetration testing.
- **Usage:** Identify and fix vulnerabilities such as injection attacks, cross-site scripting, and security misconfigurations.

5. Accessibility Testing:

- **Overview:** Use tools like Lighthouse or Axe to check your application's accessibility.
- **Usage:** Ensure compliance with accessibility standards and improve usability for users with disabilities.

6. Regression Testing:

- **Overview:** Re-run tests after each code change to ensure that new changes do not introduce new bugs or break existing functionality.
- **Usage:** Automate regression tests and integrate them into your CI/CD pipeline.

Debugging and Testing web Applications

key strategies and tools for debugging and testing web applications:

Testing:

7. Usability Testing:

- **Overview:** Have real users interact with your application and provide feedback on its usability.
- **Usage:** Identify areas for improvement in terms of user experience and interface design.

8. Performance Testing:

- **Overview:** Evaluate the speed, responsiveness, and overall performance of your application.
- **Usage:** Identify and address performance bottlenecks to improve user experience.

By integrating these debugging and testing strategies into your development workflow, you can enhance the quality, reliability, and performance of your web applications.

Best practices for maintaining and updating web applications

Maintaining and updating web applications is essential to ensure they remain secure, performant, and aligned with evolving business requirements.

Here are some best practices for maintaining and updating web applications:

1. **Regular Software Updates:**

Keep all software components, including the operating system, web server, database, and third-party libraries, up to date.

Benefits: Patch security vulnerabilities, access new features, and leverage performance improvements.

Best practices for maintaining and updating web applications

Best practices for maintaining and updating web applications:

4. Monitoring and Analytics:

Implement monitoring tools to track application performance, user behavior, and system health.

Benefits: Proactively identify and address issues, gain insights into user interactions, and make data-driven decisions.

5. Automated Testing:

Implement a robust automated testing strategy, including unit tests, integration tests, and end-to-end tests.

Benefits: Identify regressions early, ensure code reliability, and streamline the development process.

Best practices for maintaining and updating web applications

Best practices for maintaining and updating web applications:

6. Backups and Disaster Recovery:

Regularly back up data and implement a robust disaster recovery plan.

Benefits: Minimize data loss in case of emergencies, expedite recovery, and ensure business continuity.

7. Version Control:

Use version control systems (e.g., Git) to manage code changes and track the history of your application.

Benefits: Facilitate collaboration among developers, roll back to previous versions if needed, and maintain a clear codebase history.

Best practices for maintaining and updating web applications

Best practices for maintaining and updating web applications:

8. **Documentation:**

Maintain comprehensive documentation for the application, including code comments, API documentation, and system architecture.

Benefits: Aid in onboarding new developers, enable effective troubleshooting, and ensure knowledge transfer within the team.

Best practices for maintaining and updating web applications

Best practices for maintaining and updating web applications:

9. Incremental Updates:

Implement incremental updates rather than massive overhauls.

Benefits: Reduce the risk of introducing bugs, make it easier to identify issues, and allow for quicker rollback in case of problems.

By incorporating these best practices into your maintenance and updating processes, you can foster a culture of continuous improvement, reduce risks, and ensure the long-term success of your web applications.

Web Application Hosting
Domain name

Web application hosting involves making your web application accessible over the internet. The requirements for hosting a web application can vary based on the size, complexity, and specific needs of your application.

Here are some few requirements as per our need

1. Domain Name
2. Web host
3. Files to be hosted
4. Database

Web Application Hosting
Domain name

A domain name consists of two main parts:

1. **Second-Level Domain (SLD):**

   This is the main part of the domain name and is typically the unique and memorable name chosen by the owner of the domain. **For example**, in the domain name **“wap.com,"** **“wap**" is the second-level domain.

2. **Top-Level Domain (TLD):**

   This is the last part of the domain name, which often indicates the type or purpose of the website. Common generic top-level domains (gTLDs) include ".com," ".org," ".net," and country-code top-level domains (ccTLDs) like ".ug," ".kr," or ".us."

In the **wap** **“wap.com**," **".com**" is the top-level domain, and **“wap**" is the second-level domain.

Web Application Hosting
Domain name

Here are some key points about domain names:

1. **Unique Identifier:** Each domain name is unique, allowing it to serve as a distinct identifier for a specific location on the internet.
2. **URL Structure:** When combined with the appropriate protocol (usually **"http://"** or **"https://:///")** and specific resource paths, a domain name forms a complete URL used to access web pages. For example, **"https://www.wap.com/index.php"**
3. **Registration:** Domain names are registered through domain registrars, entities authorized to manage and register domain names on behalf of individuals or organizations. Registrants pay a recurring fee to maintain ownership of the domain.

Web Application Hosting
Domain name

Key points about domain names:

4. **Subdomains:** In addition to the main domain, subdomains can be created to organize and structure websites further. For instance, "blog.wap.com" and "shop.wap.com" are subdomains of the “wap.com" domain.
5. **DNS:** The domain name system (DNS) translates human-readable domain names into IP addresses, allowing computers to locate and connect to each other over the internet.

Choosing an appropriate and memorable domain name is important for online branding, accessibility, and ease of use. When users enter a domain name in a web browser, the DNS system resolves it to the corresponding IP address, enabling the browser to retrieve the requested web page from the server associated with that IP address.

Web Application Hosting
Web host

A web host is a service provider that offers the infrastructure and resources needed to make a web app accessible on the internet.

In simpler terms, a web host is like a virtual landlord for your website, providing a place for your website's files, databases, and other components to reside.

When users access your website through a web browser, the web host delivers the necessary files to their devices, allowing them to view and interact with your site.

Web Application Hosting

Web host

Example of paid and free web hosts include;

1. Paid: Bluehost, SiteGround, HostGator, A2 Hosting and InMotion Hosting etc. Bluehost. (n.d.).
2. Free:

Web Application Hosting
Web host

Free Web hosts

1. **000webhost:** Offers free hosting with PHP, MySQL, and cPanel support. Ad-supported.
2. **InfinityFree:** Provides free hosting with unlimited disk space and bandwidth., Ad-supported.
3. **AwardSpace:** Offers free hosting with PHP, MySQL, and a website builder. Ad-supported.
4. **Freehostia:** Provides free hosting with PHP, MySQL, and an easy-to-use control panel. Ad-supported.
5. **ByetHost:** Offers free hosting with PHP, MySQL, and Softaculous script installer.Ad-supported.
6. **5GBFree:** Offers free hosting with 5 GB of disk space, PHP, and MySQL support. Ad-supported.
7. **Google Cloud Free Tier:** Google Cloud Platform provides a free tier with limited resources, suitable for hosting small projects. Limited to specific services and resources.

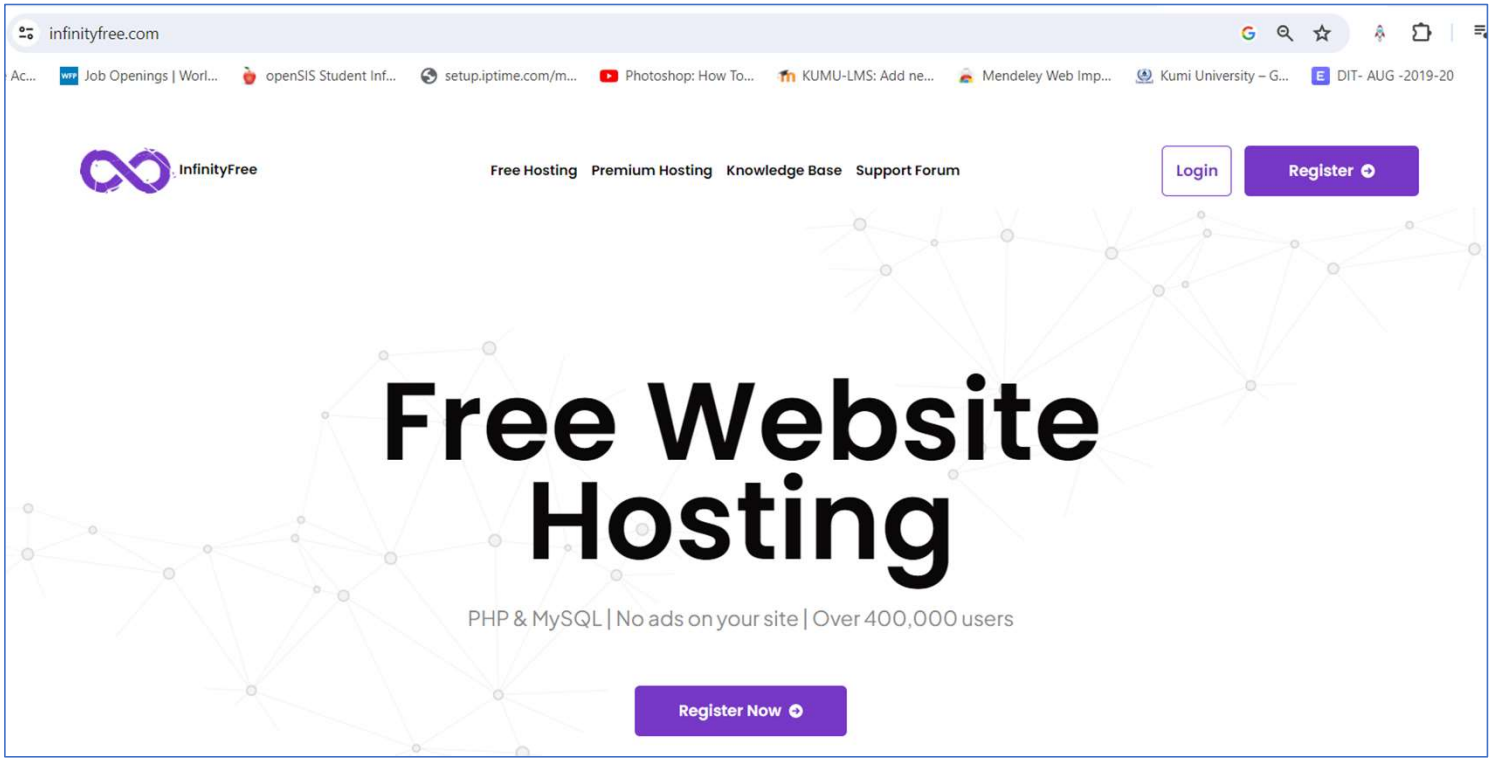
Web Application Hosting
Web host

Free Web hosts

8. **GitHub Pages:** Specifically for hosting static websites directly from a GitHub repository. Limited to static content (HTML, CSS, JavaScript) and Jekyll sites.
9. **Netlify:** Offers free hosting for static websites with continuous deployment from Git repositories. Ideal for static sites and single-page applications.
10. **Heroku (Free Plan):**Provides a free hosting plan for simple applications with limitations on resources. Suitable for small projects and experiments.

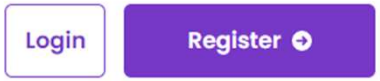
Domain Registration

Now that we have Identified some of the free domain hosting companies above, let now go a head and register a domain for ourselves on https://www.infinityfree.com/ or



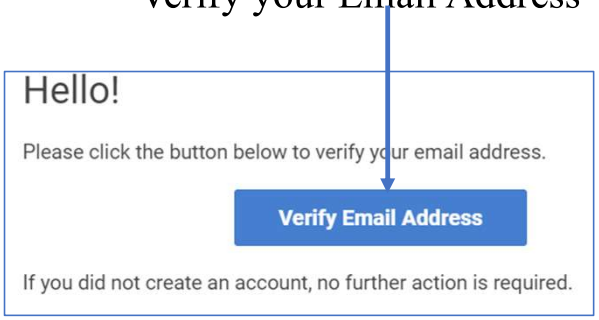
Domain Registration
Creating user account with Host

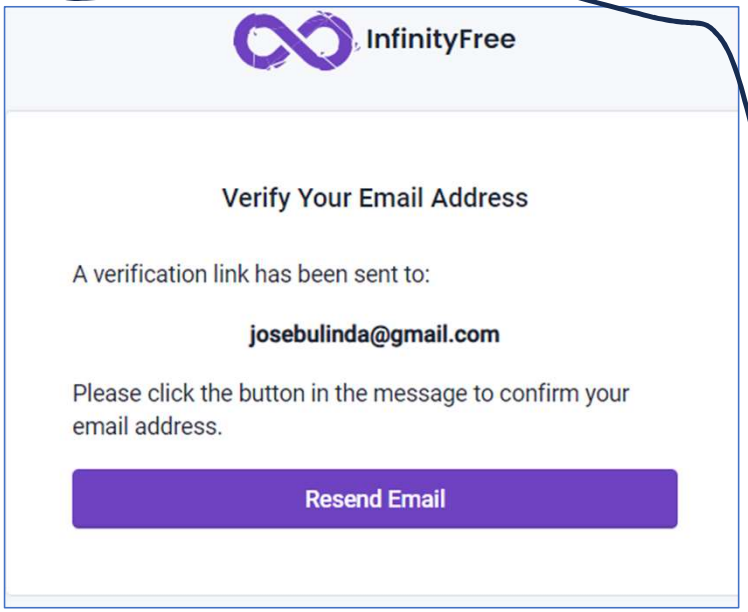
2. Click register button

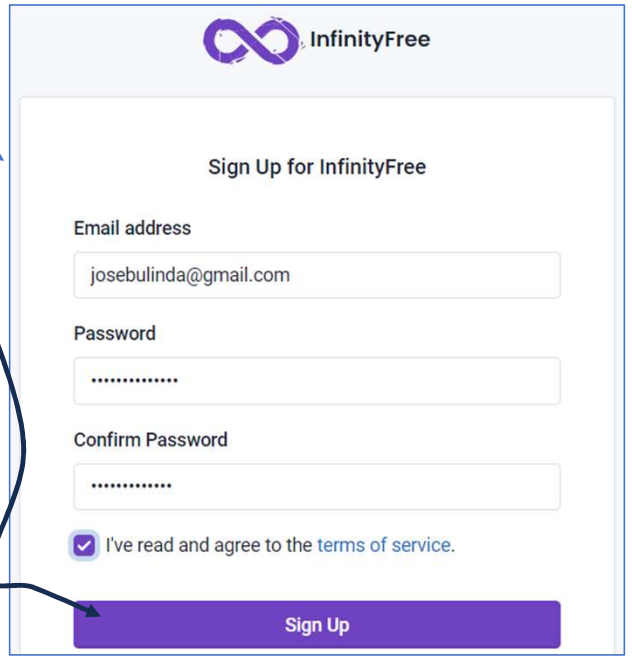


3. Fill in the registration form
4. Click Sign up
5. Open you email to

Verify your Email Address

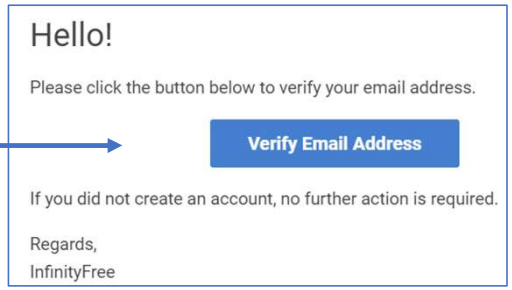




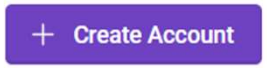


Domain Registration
Creating user account with Host+
Selecting Hosting plan

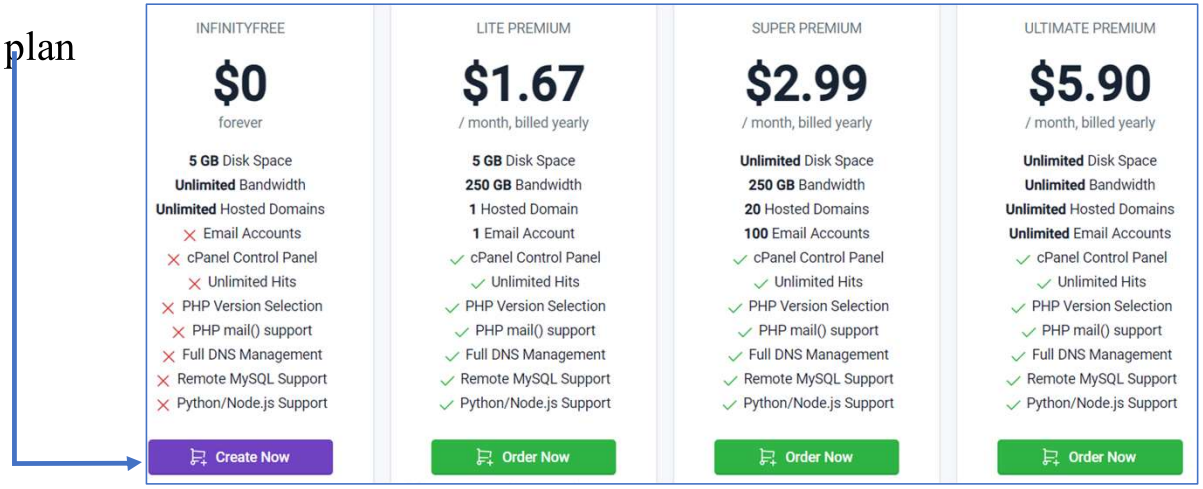
6. Open your email then click Verify Email Address



7. Click Create Account

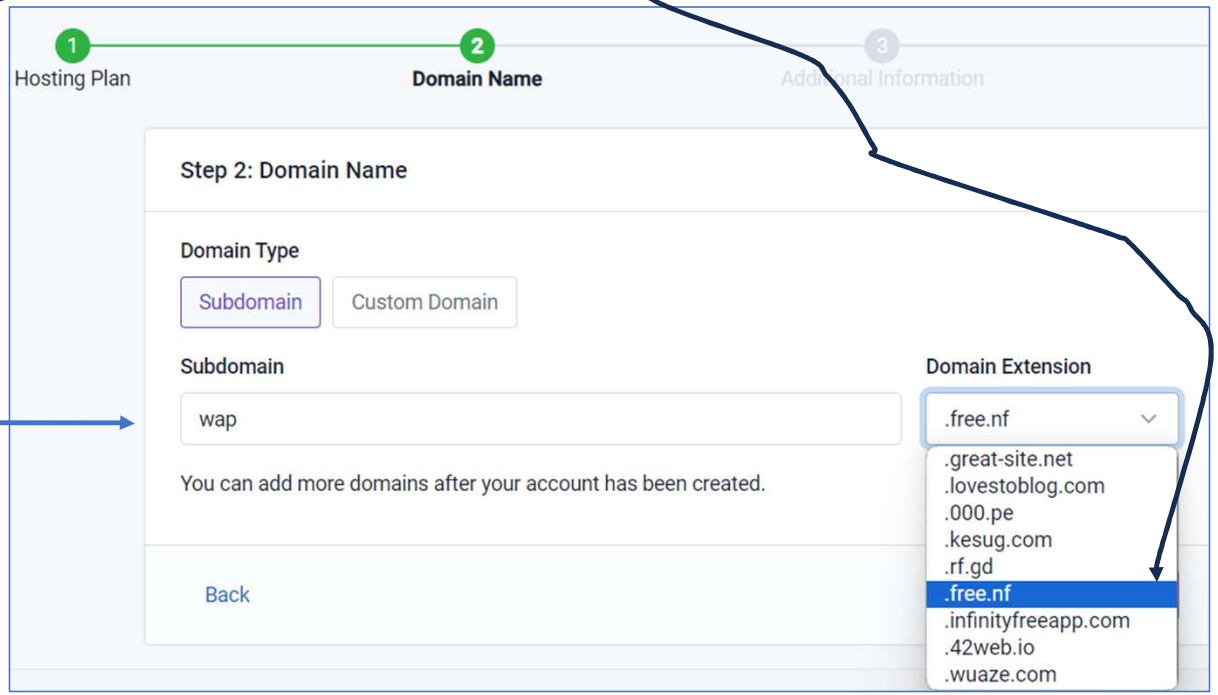


8. Choose free hosting plan



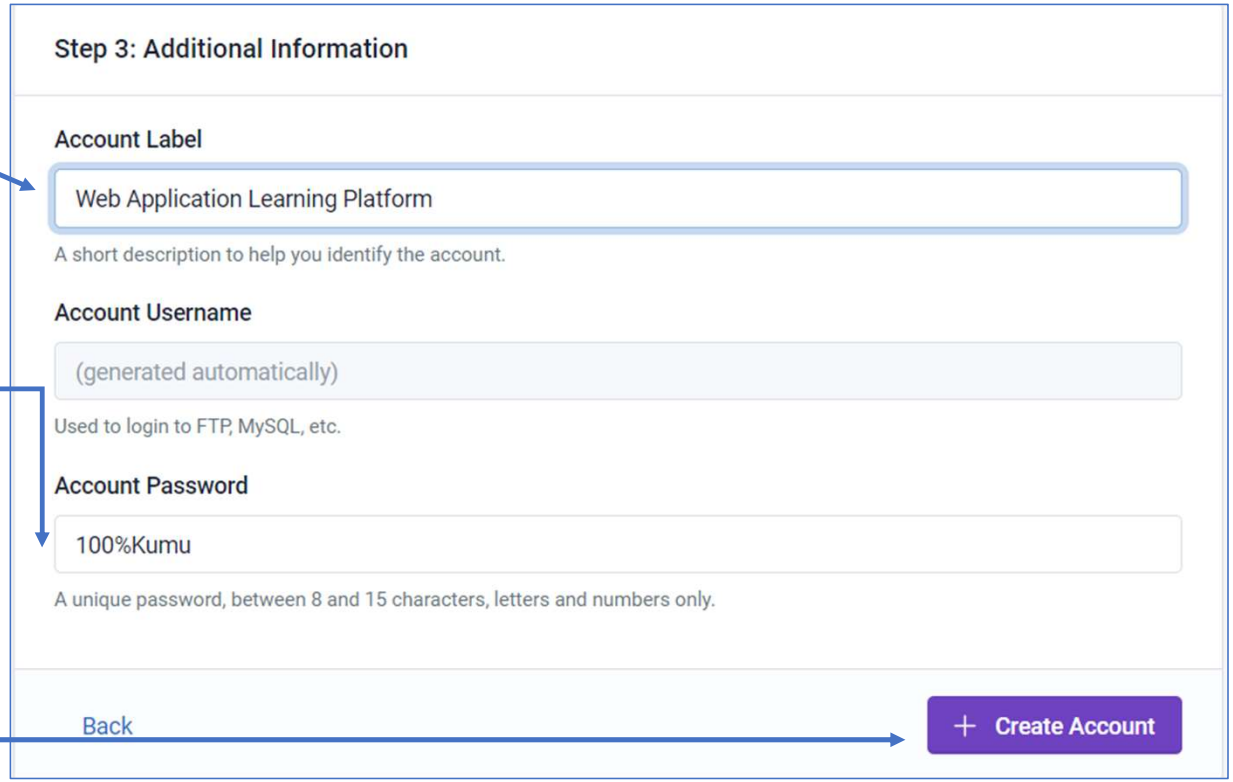
Domain Registration
Creating user account with Host+
Selecting Hosting plan

9. Enter subdomain name e.g **wap** in the text box
10. Select Domain extension e.g **free.nf**

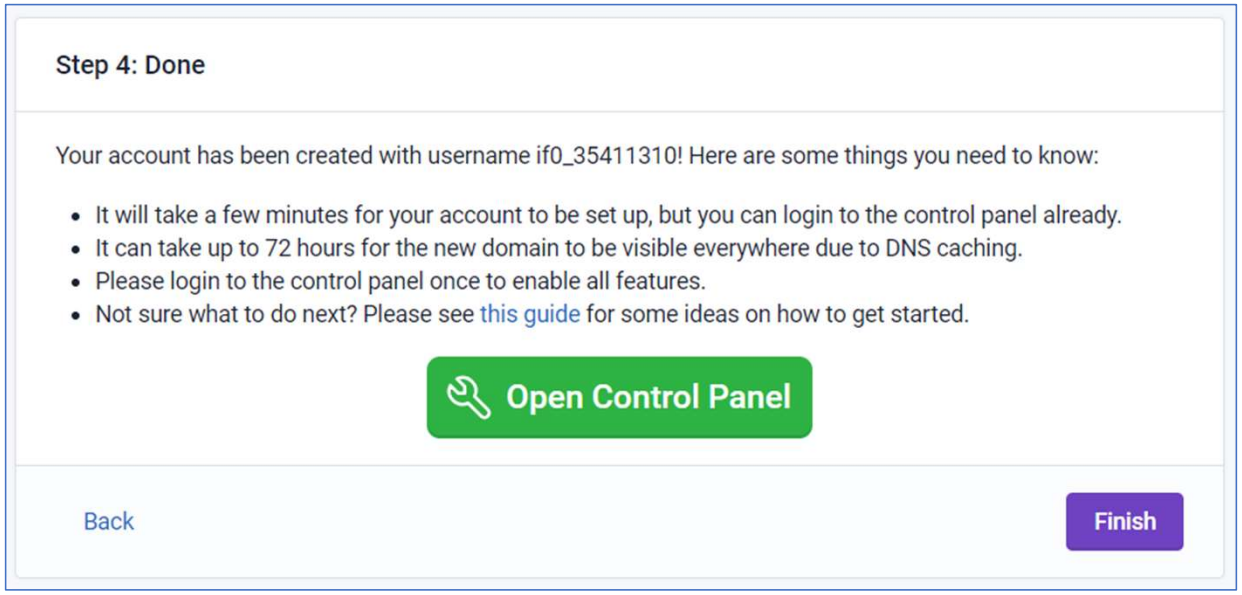


Domain Registration
Creating user account with Host+
Selecting Hosting plan

11. Click Check a Availability
12. Enter account Label
13. Enter account password
14. Click Create Account



Domain Registration
Creating user account with Host+
Selecting Hosting plan

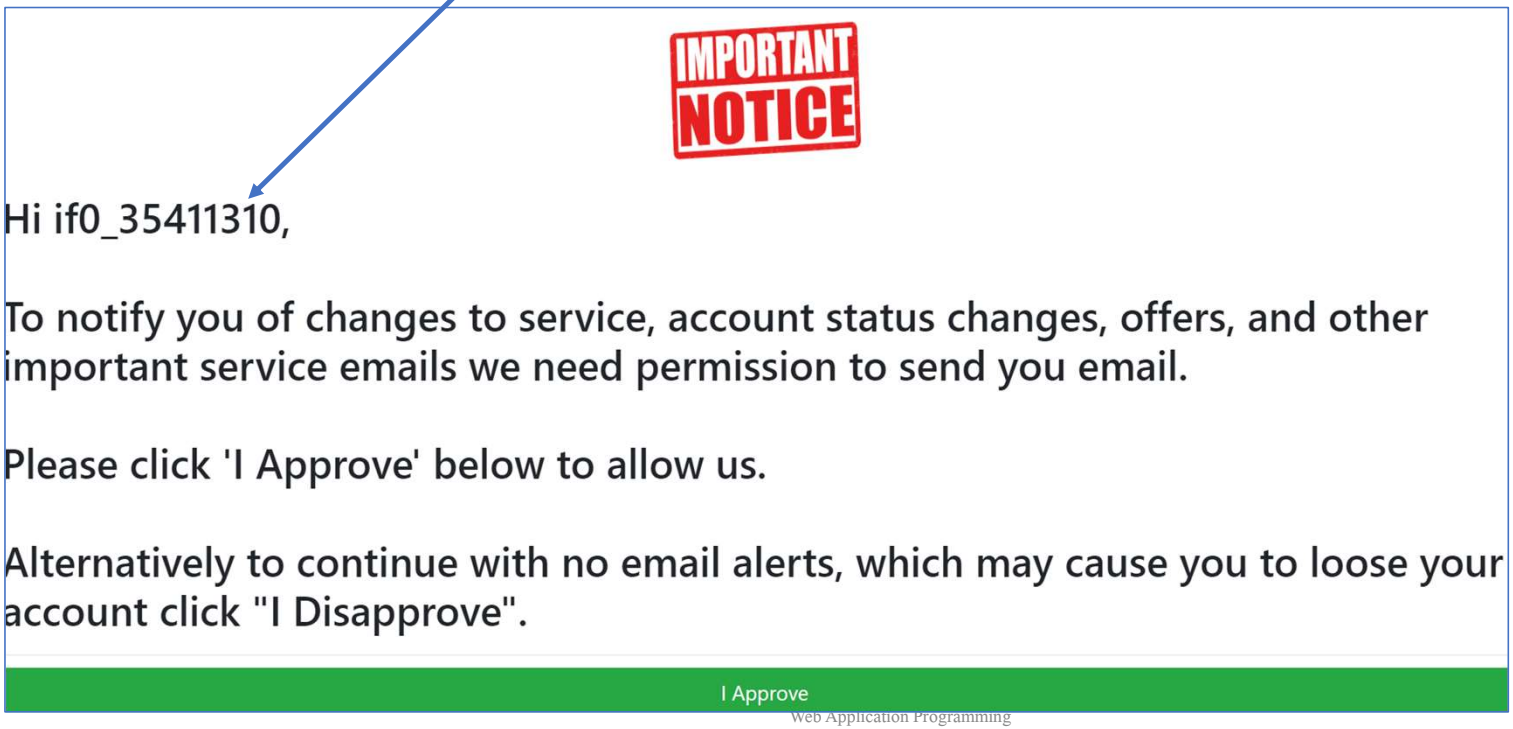


Clicking finished means you have created the domain name successfully

However let click Open control Panel to create the database upload our wap files

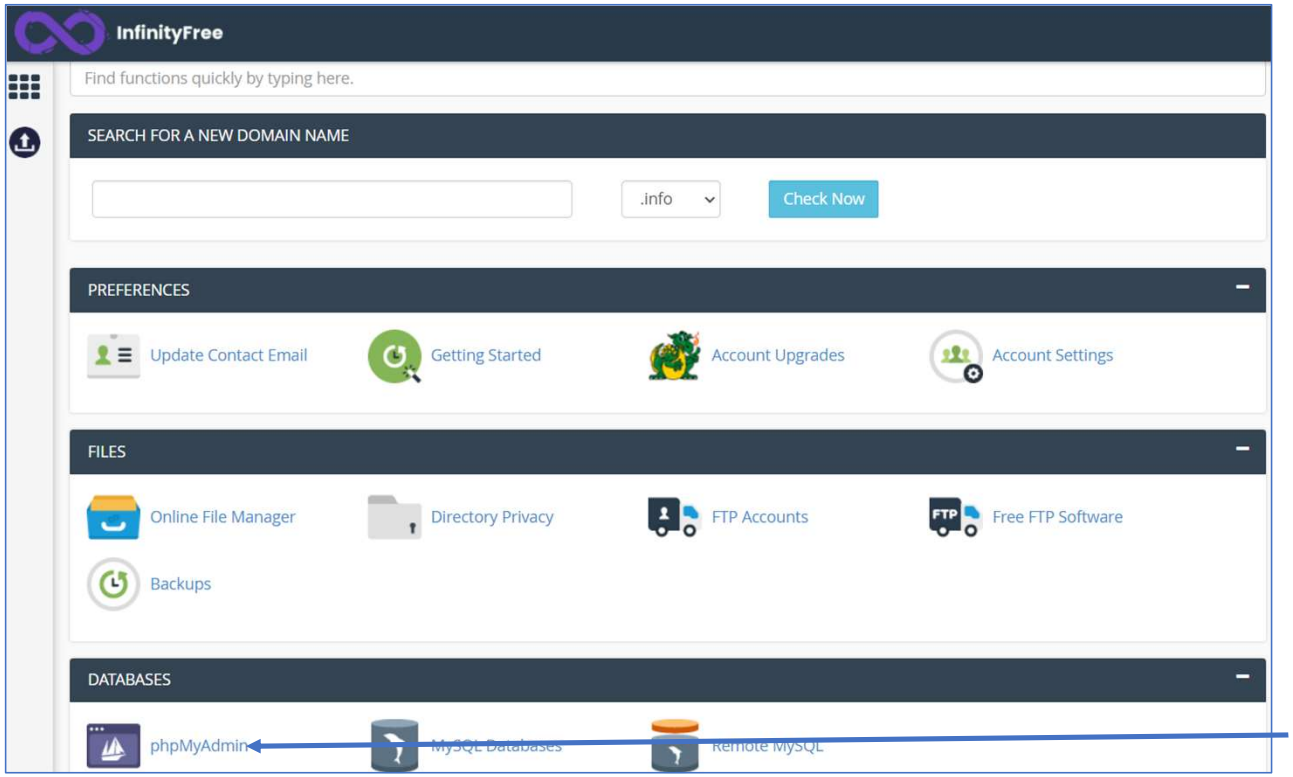
Domain Registration
Creating user account with Host+
Selecting Hosting plan

Take note of your username: **if0_35411310** displaced on the notice page. Click **I Approve**



Domain Registration
Creating user account with Host+
Selecting Hosting plan

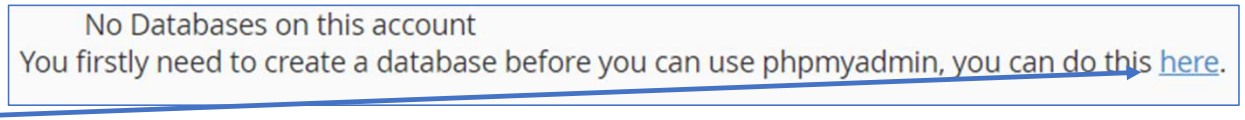
See the cPanel items-the control point of your domain



We can now create the database within which we will import the offline database, by clicking **phpMyAdmin** option

Domain Registration
Creating user account with Host+
Creating a database

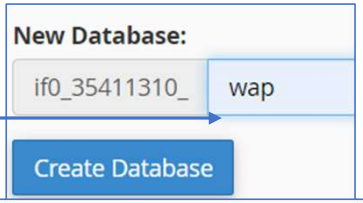
17. On clicking phpMyAdmin, you see



18. Please go ahead and click **here**

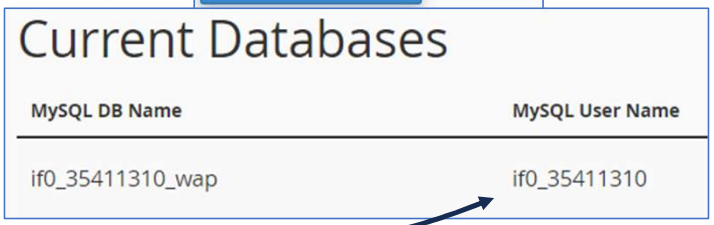
19. Enter database name e.g **wap** in the New Database text box

20. **Click create Database**



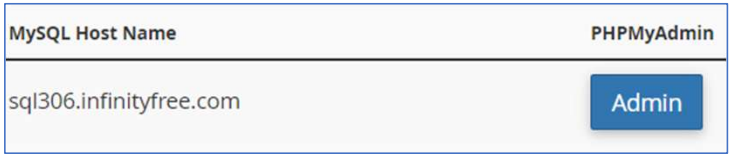
Database created; Note four things

1. **Database name:** if0_35411310_wap
2. **User name :** if0_35411310
3. **Password:** Your panel password: 100%Jonet20**
4. **And the Host:** sql306.infinityfree.com

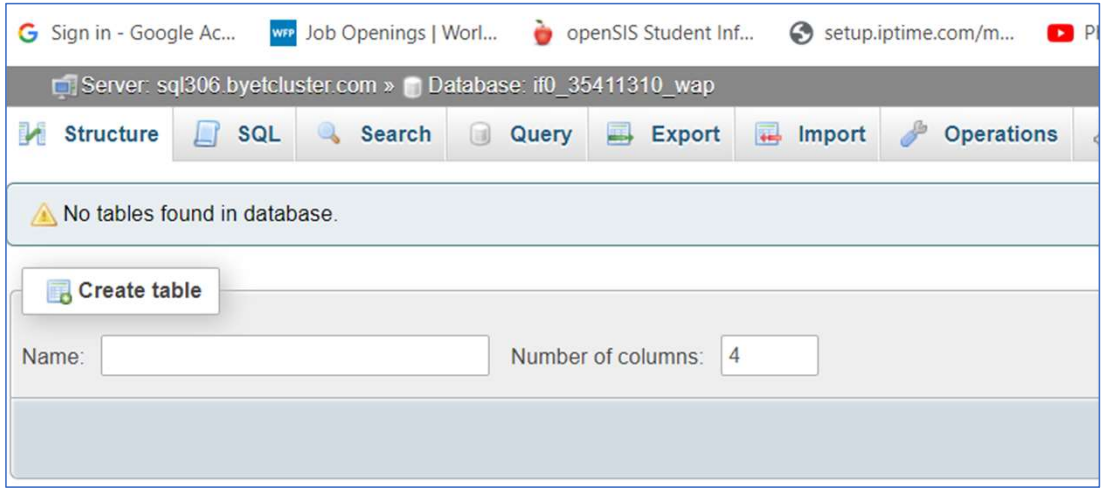


Domain Registration
Creating user account with Host+
Accessing phpMyAdmin

21. Click Admin to Access phpMyAdmin section



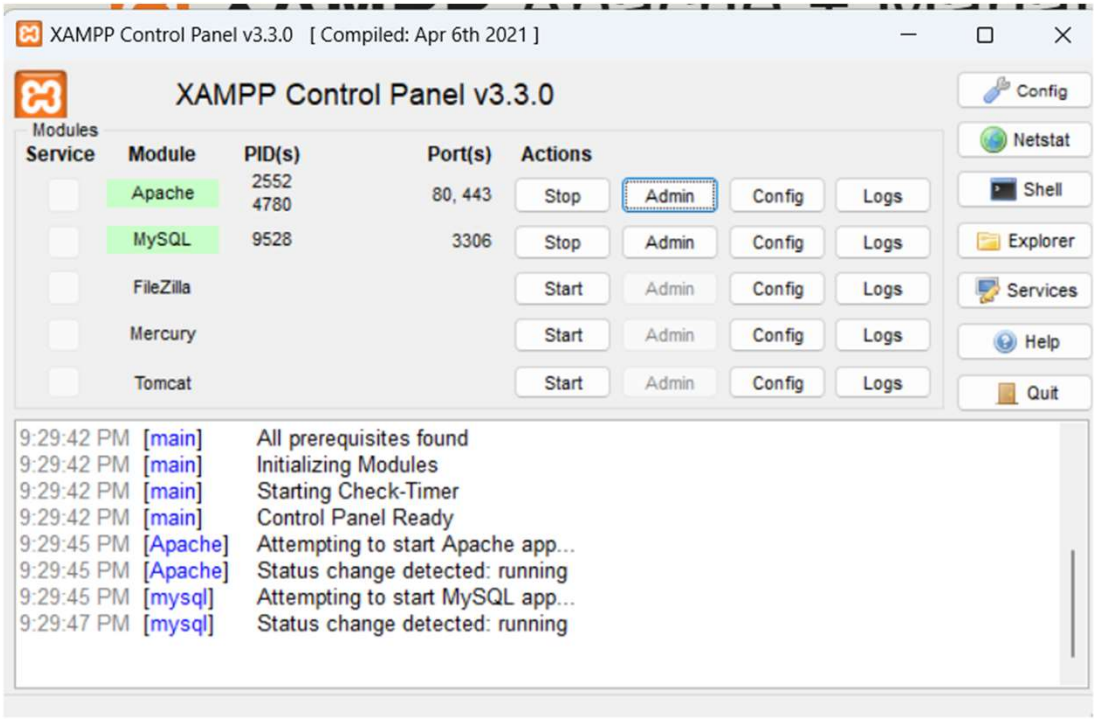
22. Finally ready to import the database



Now we need to export our local database so that we import it online

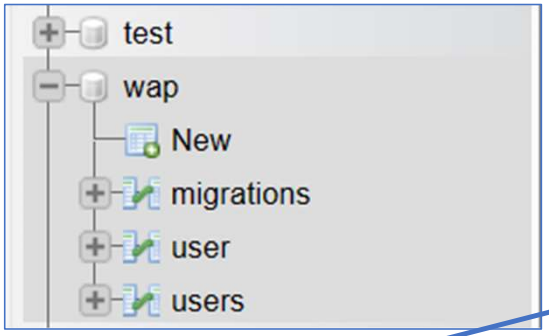
Export wap database to the online database

Start you local XAMPP, start Apache and MySQL



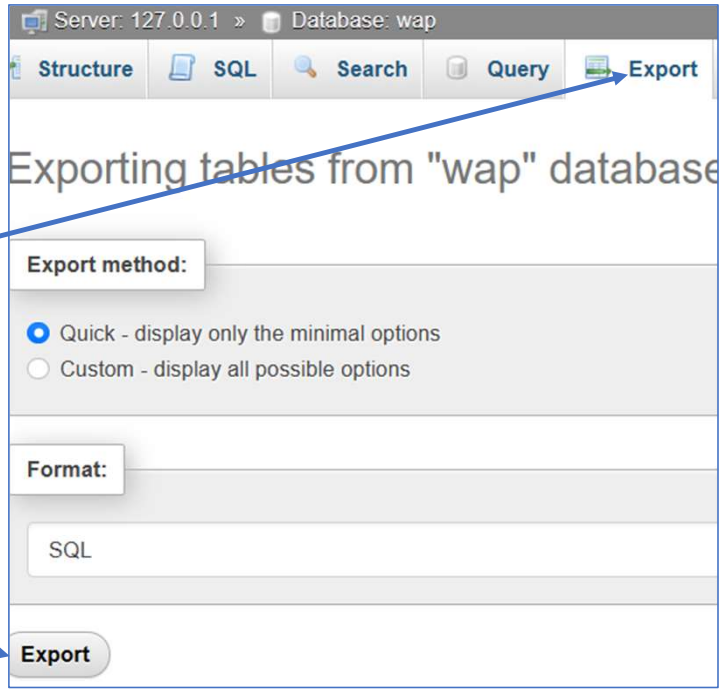
Export wap database to the online database

1. Click wap database or the database you want o export



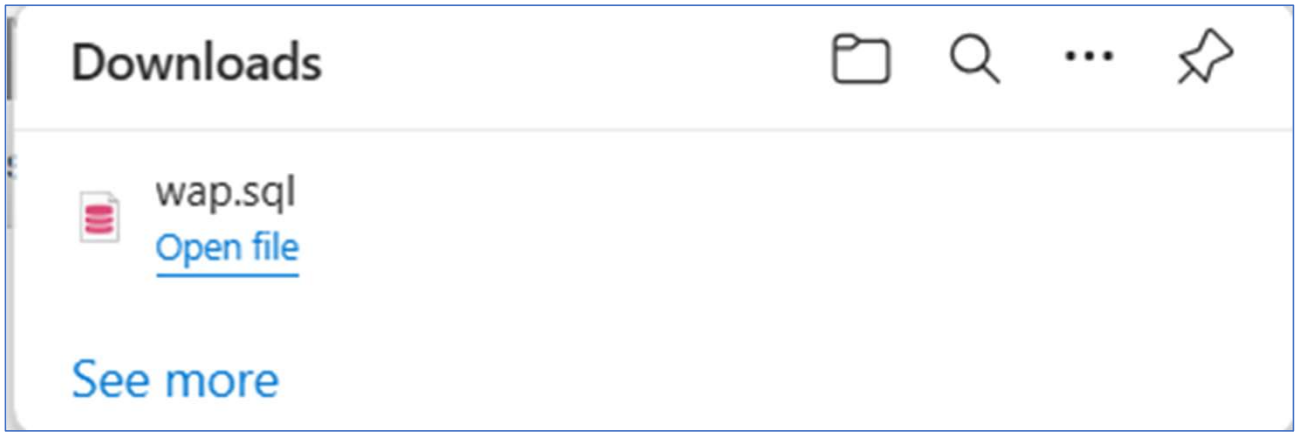
1. Click Export

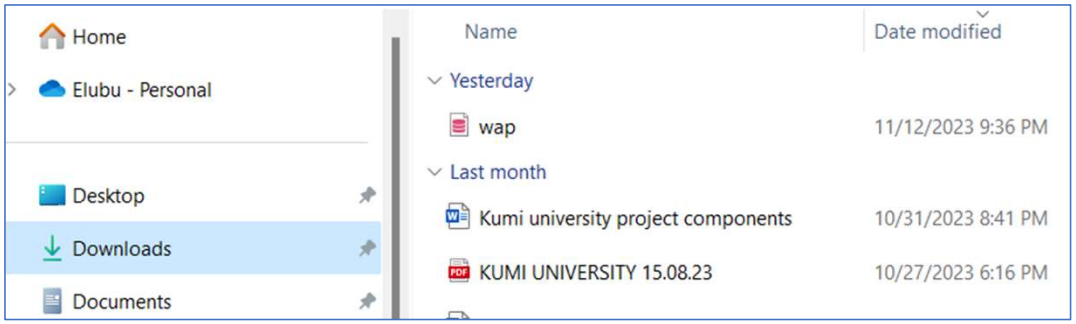
2. Click Export button at the bottom



Export wap database to the online database

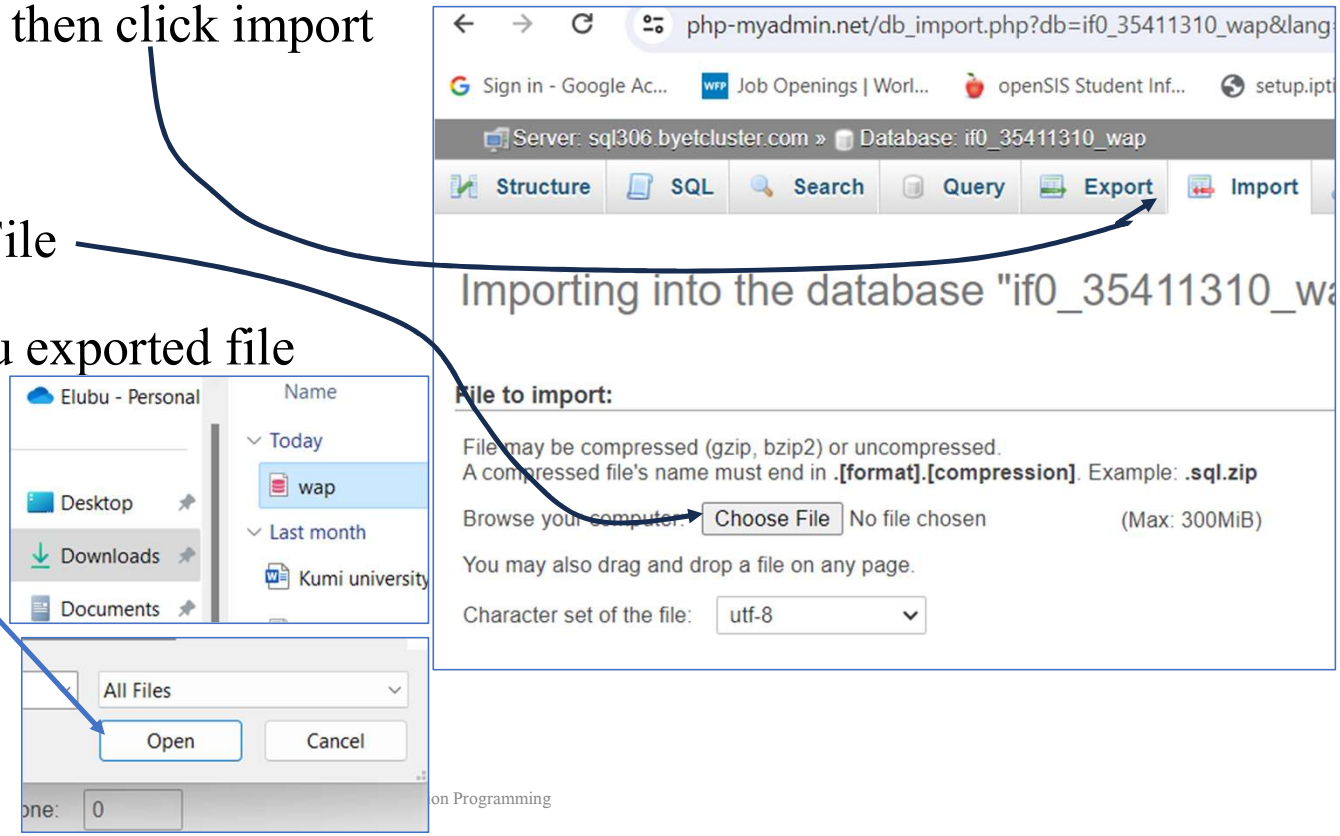
3. Locate where you exported database is e.g Downloads folder





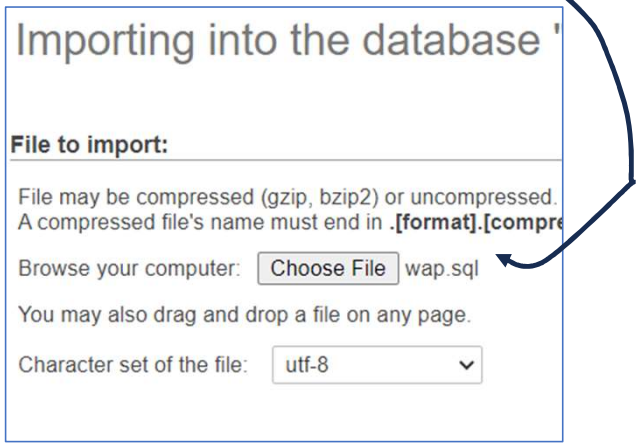
Import wap database to the online database

1. Go back online then click import
2. Click Choose File
3. Navigate to you exported file
4. Click open

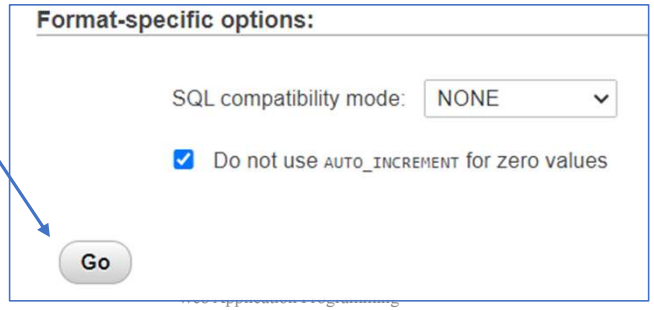


Import wap database to the online database

5. Note that **wap.sql** is seen next to Choose File button



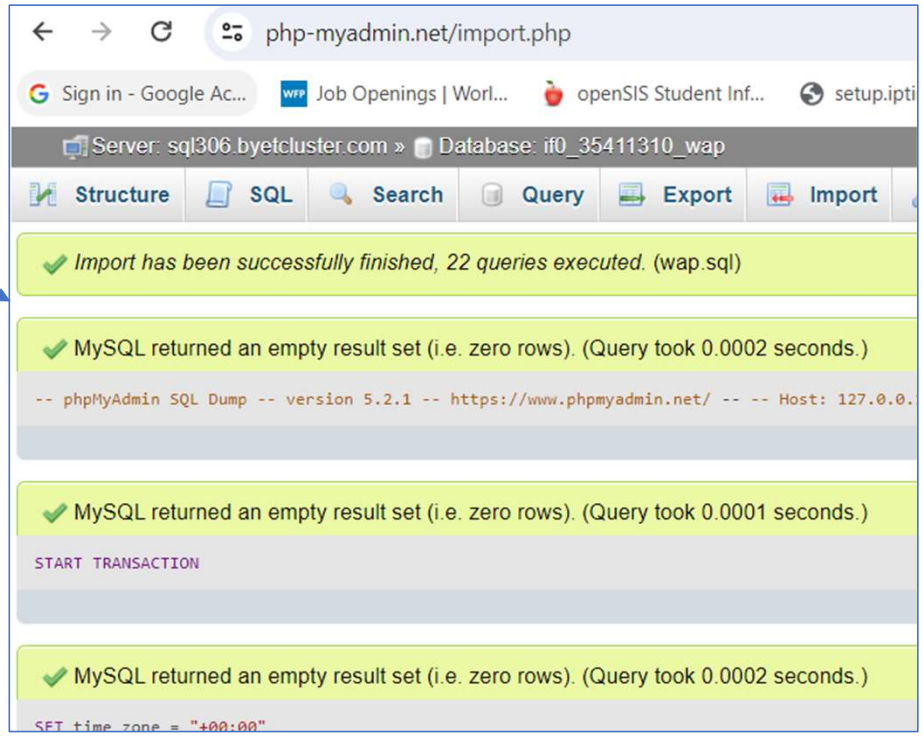
6. Now scroll down and click **Go**



Import wap database to the online database

If you can see this,

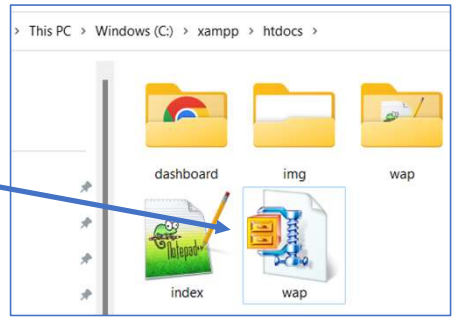
Then you are successful



Now we need to upload wap folder files to our online server

Upload Offline files to the web host online

1. Locate you offline files and zip the folder e.g.



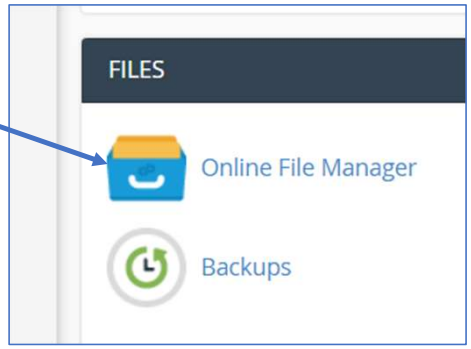
2. Go back to your cPanel

http://cpanel.infinityfree.com/index.php

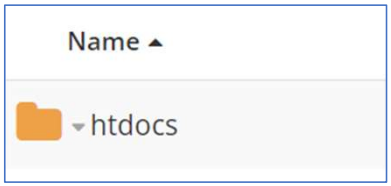
Sign in using the details sent to your email address

Upload Offline files to the web host online

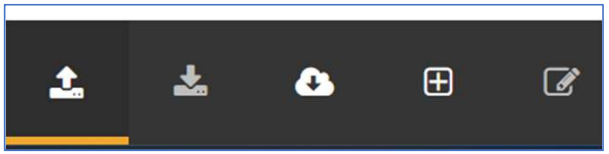
3. Click on Online File Manager



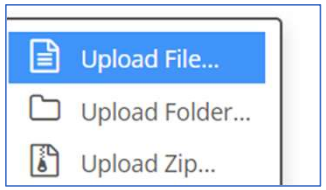
4. Open **htdocs** folder



5. Click on Upload Icon found at the bottom of the page



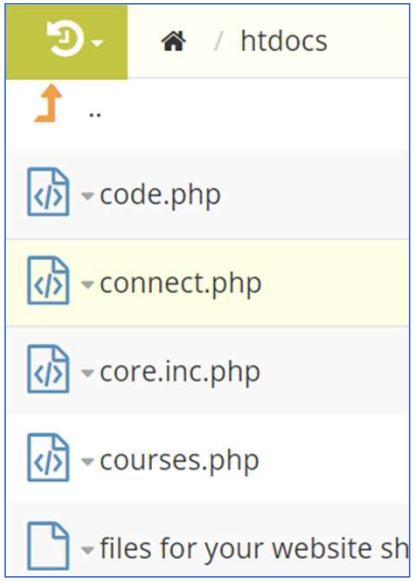
6. Click Upload Files



7. Select the files then click open

Upload Offline files to the web host online

Open connect.php file and edit your connections details

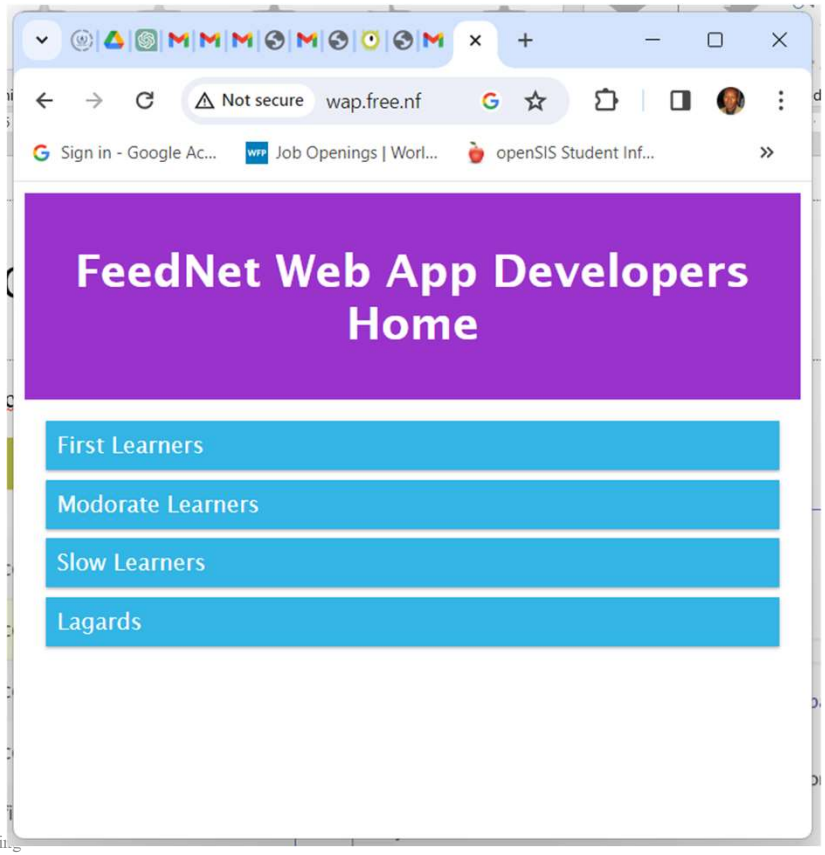



```php
<?php

    $server = "sql306.infinityfree.com";
    $username = "if0_35411310";
    $password = "fw63d2h4";
    $database = "if0_35411310_wap";

    $conn = new mysqli($server, $username, $password, $database);

    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);

    }
```

Now visit your Domain: wap.free.nf

Upload Offline files to the web host online

Note that **wap.free.nf** is now live online

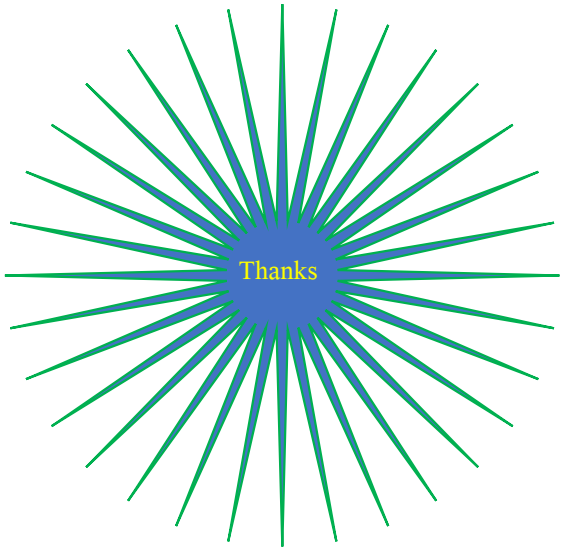


Summary

Summary

1. Web Application Deployment and Maintenance(
   i. Looked a list of hosting options
   ii. deployment strategies,
   iii. Looked at ways of Debugging and testing web applications
   iv. Discussed best practices for maintaining and updating web applications)
   v. Registered with with a hosting company, registerd a subdomain(wap.free.nf) and hosted the files.

Thank you for Listening



References

The PHP framework for web artisans. Laravel. (n.d.). https://laravel.com/docs/10.x#getting-started-on-windows

Bluehost. (n.d.). Web Hosting Services. Retrieved from https://www.bluehost.com/