RESEARCH ON DEPLOYMENT STRATEGIES AND HOSTING PLATFORMS

Deployment Strategies

**Traditional Server Hosting**: In this approach, you have to rent a physical or virtual server from a hosting provider, install your application on it, and manage everything yourself.

**Pros**: Full control over the server, more customization options.

**Cons**: Requires more technical expertise, responsibility for server maintenance and security.

**Cloud Hosting**: In cloud hosting, your application is hosted on a network of remote servers, and you can scale resources as needed.

**Pros**: Scalability, high availability, and you only pay for what you use.

**Cons**: Can get expensive, less control over the underlying infrastructure.

**Serverless Architecture**: In this approach, you don't manage servers; instead, you deploy functions that execute in response to events (like an HTTP request).

**Pros**: No need to manage servers, automatic scaling.

**Cons**: Can be complex to debug, might have latency issues.

**Containerization (e.g., Docker)**: Here, your application and its dependencies are packaged into a container that can run consistently across any environment.

**Pros**: Consistency across environments, easier to scale and manage.

**Cons**: Requires learning about container management, can be complex to set up initially.

Hosting Platforms

**AWS (Amazon Web Services)**: A popular cloud platform with a wide range of services.

**Pros**: Highly scalable, lots of services, widely used.

**Cons**: Can be expensive, has a steep learning curve.

**Google Cloud Platform**: Similar to AWS, it offers cloud computing services and is great for scalability.

**Pros**: Integrated with Google services, powerful tools for data analysis.

**Cons**: Learning curve, can be expensive.

**Heroku**: This is a platform as a service (PaaS) that simplifies app deployment, especially for small to medium-sized projects.

**Pros**: Easy to use, good for beginners, free tier available.

**Cons**: Less control over server settings, limited free tier.

**DigitalOcean**: A cloud computing platform known for simplicity and affordability.

**Pros**: Easy to set up, good for small projects, affordable.

**Cons**: Fewer advanced features compared to AWS and Google Cloud.

Choosing a Deployment Method

* Project Requirements: Consider the scale of your project. If it’s a small project, something like Heroku or DigitalOcean might be suitable.
* Budget: If you're looking for something free or cheap, Heroku's free tier or DigitalOcean’s low-cost plans are good options.
* Scalability: If you plan to grow the application, AWS or Google Cloud might be better, but they come with higher costs.
* Technical Expertise: If you’re new to deployment, Heroku or DigitalOcean is more beginner-friendly.

Manual vs. Automated Deployment:

**Manual Deployment**: You upload your code to the server and manually configure everything.

**Pros**: More control, better for learning the basics.

**Cons**: Time-consuming, error-prone.

**Automated Deployment (CI/CD)**: Continuous Integration/Continuous Deployment pipelines automatically build, test, and deploy your code whenever you push changes to a repository.

**Pros**: Saves time, reduces human errors, enables faster iteration.

**Cons**: Requires setting up CI/CD pipelines and learning how they work.

Setting up the Live Server Environment

**Choose Your Hosting Platform**: Let’s say you pick Heroku for its ease of use.

Set Up the Server:

For Heroku, you will need to install the Heroku CLI on your computer.

Create a new application on Heroku and connect it to your GitHub repository.

Configure your server with the necessary software like Node.js and a MySQL database.

Deploying the Expense Tracker Application

Prepare the Application:

* Optimize your code, remove unnecessary comments, minify your CSS/JS files.
* Set up environment variables, for example, storing your database URL and secret keys.
* Handle production settings, configure your app for production (e.g, use a production database).

Deploy the Frontend:

* Push your HTML, CSS, and JavaScript files to the server.
* Ensure they are properly linked and work in the live environment.

Deploy the Backend:

* Deploy your Node.js server with Express.js to the live environment.
* Make sure your routes are working, and your server can connect to the database.

Database Connections:

* Make sure your backend can connect to the MySQL database on the live server.
* Set up the necessary environment variables to securely store your database credentials.

Testing and Verification

Test the Application:

* Visit your live app and test all functionalities (login, adding expenses, viewing expenses, etc.).
* Look for any bugs or issues that weren’t present in the local environment.

Cross-Browser Testing:

* Test your app in different browsers (Chrome, Firefox, Safari) and devices (desktop, tablet, mobile) to ensure it works everywhere.

Verify Data Integrity:

* Ensure that data is being saved and retrieved correctly.
* Check security measures like data encryption, HTTPS.