First, let's start with the user who wants to access your website, www.foobar.com. When a user types the URL in their web browser, their computer initiates a request to the DNS (Domain Name System) server to resolve the domain name. The DNS server translates the domain name, www.foobar.com, into an IP address, in this case, 8.8.8.8, which represents the server that hosts your website.

Now, let's discuss the components of the infrastructure:

1. Server: A server is a physical or virtual machine that hosts and serves the website's files and processes user requests. In this case, we have one server that performs multiple roles.
2. Web Server (Nginx): The web server's role is to handle incoming HTTP requests from users and serve the corresponding web pages. We'll use Nginx, a popular web server software, for this purpose. Nginx receives requests from users and communicates with the application server to process and generate the appropriate response.
3. Application Server: The application server hosts the application's codebase. It is responsible for executing the application's logic, handling dynamic content generation, and communicating with the database. In our case, we'll assume a LAMP stack, where the "A" stands for Apache, which serves as the application server and handles the PHP code.
4. Application Files (Codebase): These are the files that constitute the website's application logic and functionality. They are stored on the server and are executed by the application server when a request is received.
5. Database (MySQL): The database stores and manages the website's data. MySQL is a popular open-source relational database management system (RDBMS) that we'll use in this infrastructure. It allows us to store and retrieve data efficiently.

The DNS record for www.foobar.com would be an A record. An A record maps a domain name to an IPv4 address. In this case, the A record for www.foobar.com would be configured to point to the server's IP address, 8.8.8.8.

To communicate with the user's computer, the server uses the HTTP protocol. When a user sends an HTTP request to www.foobar.com, the server receives the request and responds with the appropriate web page or data.

Now, let's discuss the issues with this infrastructure:

- Single Point of Failure (SPOF): Since we have only one server, if it fails or experiences any issues, the entire website will go down. There is no redundancy or backup server to handle requests in case of failure.
- Downtime during maintenance: When performing maintenance tasks such as deploying new code or restarting the web server, the website would experience downtime. During this period, users won't be able to access the website.
- Limited scalability: With only one server, the infrastructure cannot easily handle a significant increase in incoming traffic. If there is a sudden surge in visitors, the server may become overloaded, resulting in slow response times or even a complete outage.

To address these issues, you could consider implementing solutions like load balancing, horizontal scaling, and redundancy by introducing multiple servers or cloud-based infrastructure. Additionally, utilizing caching mechanisms, content delivery networks (CDNs), and optimizing code and database queries can help improve performance and scalability.