**Basic Flow for Website**

**A. Planning Phase**

1. Define the main purpose of website.
2. Find the target audience.
3. Create flow diagram, site-map to visualize the structure & layout of website.
4. Decide different features of website and tech-stack requirements.
5. A single website may use different frontend and backend technology at same time to fulfil certain tasks. [Keep this in mind]

**B. UI/UX Design & Frontend Development**

1. Design a good UI layout and structure.
2. Describe how the website should look according to target audience.
3. Choose the framework which you want to use for your frontend developer. [React, Angular, Vue, Next or any other]
4. You can choose any number of frameworks according to your requirements.
5. You can choose angular and react or any other in same website if it meets your requirements.

**C. Backend Designing and Development**

1. Divide your website features and structures & decide which backend language you should choose for which feature.
2. It is common that a single website is using different backend language for different features.

As different features can have different requirements and a single backend language can’t possibly have all functionality.

1. Some language and their common use cases.
2. **Node.JS, Express.JS, Next.JS**

Simple Web Development

API Handling

Real Time Applications

Chat Applications

1. **Python & Django**

Data Intensive Applications

Complex Designing

High Computation

Creating Payment and Trading Websites

Machine Learning Tasks

Deep Learning Based Tasks

With use of Libraries like Pandas, NumPy, TensorFlow, PyTorch etc

1. **Java & Spring Boot and C# & ASP.NET**

High Computation

Enterprise Level Websites

Large Scale Websites

Robust Ecosystem

Creating Payment and Trading Websites

Integration between different other existing systems

1. **C++ & Rust**

Performance Critical Operations

Low Level Requirements

**D. Choosing and Setup Database**

1. Like Backend Languages, Databases and DB Languages have to be chose based on feature you want to implement.
2. It is common for a website to use different Databases and DB Languages at same time depending upon the use case.
3. Some Common DB Languages and Their Use Case.
4. **SQL**

RDBMS

High Performance

Structured Data

Complex Design

1. **PL/SQL**

Extension of SQL

Complex Data Manipulations

Implementing Complex Business Logic

Stored Procedures and Triggers

1. **NoSQL – MongoDB & Mongoose**

Querying and manipulating in NoSQL databases like Cassandra, Couchbase & MongoDB

Retrieving Documents from Document based or Key-Value based Stores.

1. **GraphQL**

Building API that allow Clients to fetch exactly the required data

Optimize Data Fetching – By reducing over and under-fetching

1. **XQuery**

Querying and Transforming XML data optimally

Made to XML Databases and stores

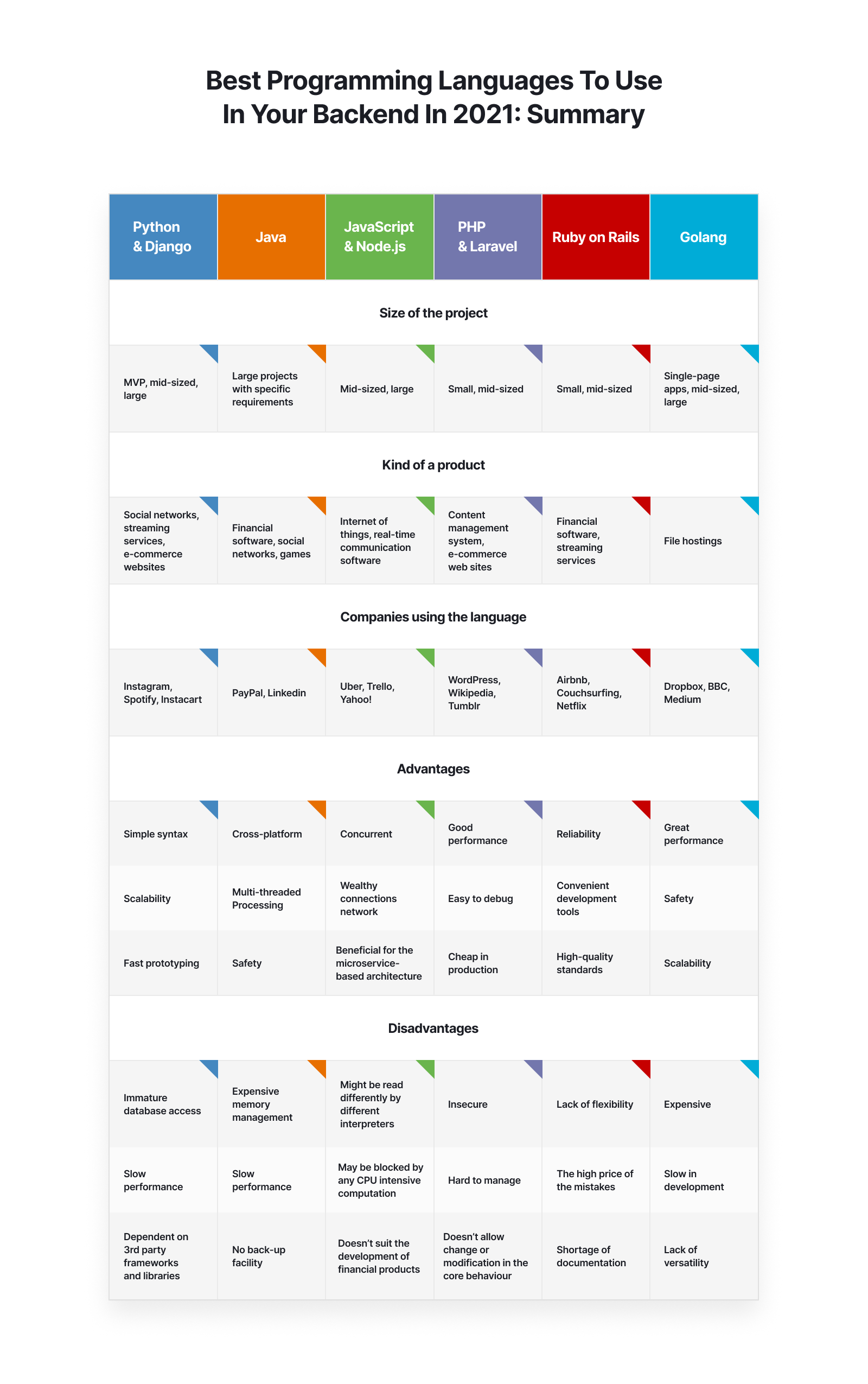
1. **Cypher**

Querying and analysing relationships and patterns in graph databases like Neo4j

Made for Querying into Graph DB

1. **LINQ [Language Integrated Query] – For C# & .NET**

Querying collections, databases, XML, and other data sources using a unified syntax within .NET languages like C#.



**E. Integration and API Development**

1. Create APIs to enable communication between the frontend and backend components.
2. Integrate third-party services or APIs if needed (e.g., payment gateways, social media integration).
3. You can Integrate any other website or service’s API if needed or required. [e.g., Spotify’s API for Music or any other.]

**F. Setup Testing Server**

1. Up till now, we have used only development server most of the times which is our localhost.
2. Now, as our development work is completed, we have to test our website. Thus, have to setup a **Testing Server**, which will test our website by different ways like sending multiple requests, testing different features and helps in analysing how our website responds to heavy traffic.

**G. Deployment and Hosting Solution -- IMPORTANT**

1. Register a Domain Name
2. Setup a **Production Server** – A VPS, A Cloud Server, or A Shared Hosting Service
3. Setup FTP Service
4. FTP Upload Service [If required]
5. DNS Configuration with Domain Name & IP.
6. Setup Email & SMTP Service

**H. Security & Performance Measures**

1. Implement security measures to protect against common web vulnerabilities (e.g., XSS, CSRF, SQL injection).
2. Use HTTPS to encrypt data transmission.
3. Optimize the website for faster loading times and better user experience.
4. Implement caching strategies, minimize HTTP requests, and optimize images and assets.
5. Implement different Optimization Techniques – Use Load Balancers, Vertical Scaling, Horizontal Scaling and Many more

**I. Maintenance & Updates**

1. Regularly update and maintain the website to fix bugs and improve performance.
2. Continuously monitor user feedback and make necessary improvements.

***IMPORTANT NOTE*** *- “This isn’t the Hard-Coded Path to follow, it is just an outline and generalized approach which will change according to requirement and implementation of particular website and use cases.”*