Web Application:

Part 1. Frontend:

The visual part of any website is done in HTML and CSS, while the interactive part of a website is mostly done in JavaScript (or TypeScript sometimes). Of course we could build a website with vanilla CSS and JS, but it is very inefficient with low scalability. There comes the Web Frameworks and the CSS Frameworks. We are only considering web frameworks for JS here because JS is specifically created for web development. The most commonly using JS Framework are Angular, React and Vue.

Part 1.1: Angular JS – suitable for enterprise-scale web app

Pros	Cons
- Unit testable: this provides the developer convenience when developing - two-way binding: reduce development time - MVC architecture: high quality code, reusability, maintainability - Typescript: cleaner code and better scalability - Directives: allowed developers to customize behaviors to the DOM), dynamic and rich content	- Limited SEO options: Webapp had low chance to be found on the top page in google - Steep learning curve: requires a lot of studies before can make a simple app - Real DOM: easy to produce bugs

Part 1.2: React JS – suitable for lightweight app, the Most Popular out of three

Pros	Cons
- Virtual DOM: makes code readable, fast to render the page, - Cross platform: supports apps in all browsers - Reusable Components: reuse the components saves time - Single direction data flow: predictable behavior and stable code - Easy to learn and use - JS Library: there are many great libraries that can be handy for various purpose - Test: very easy to test - Use with Redux:	- Large size of dependencies: causes trouble when dependencies deprecated - High pace in update: new features continuously being added and old features got replaced, needs to maintain the code periodically

Part 1.3: Vue JS – suitable for lightweight app

Pros	Cons
- Virtual DOM: makes code readable, fast to render the page, - Cross platform: supports in many browsers - Reusable Components: reuse the components saves time - Two-way data binding: - Easy to learn and use - CLI version: new GUI, easier webpack configuration	- Small Community: limited support online - Over Flexibility: over-complicate project, cause irregularities in code, which delay the development

Part 2 – Backend

1. Django (Python) with tox

Pros	Cons
- Extendable and Scalable: Native ORM layer for handling database access, sessions, routing, and multi-language support; - Secure: Django includes prevention of common attacks like Cross-site request forgery (CSRF) and SQL Injections; - Complete Built-in Admin panel.	- Potentially leads to slow website due to the ORM combined with Python.

2. Flask (Python) with Flask-Testing

Pros	Cons
- Fast: minimalistic design that gives more freedom in development; - Easy to learn: clear and concise ideology; - Integration with database toolkits like SQLAlchemy and NoSQL, databases like MongoDB, DynamoDB, etc.	- Server is single-threaded: slow when serving more than one request.

3. Ruby on Rails (Ruby) with Minitest

Pros	Cons
- Large, vocal community - Security: Cross-Site Scripting, SQL Injection, Cross-Site Request Forgery, Insecure Direct Object Reference or Forceful Browsing are supported; - High development speed: many free open-sourced libraries supported.	- Lack of flexibility: hard dependency between components and modules, customization of APPs with specific functionality can be challenging; - Lower popularity during recent years.

4. Laravel (PHP) with Artisan

Pros	Cons

- ORM support: Relationships and mapping of the database is easy to handle; - Fast execution of the web application; - Easy to use (easy to learn).	- Lightweight: May be hard to setup and integrate third-party tools for large and very custom websites. - Legacy systems not easily transferred. Some companies will stay with Zend or Symfony, because it would make too much effort to build it again with LV.

5. Spring Boot (Java) with WebMvcTest

Pros	Cons
- Simplified and version conflict free dependency management through the starter POMs; - Fast setup and run standalone, web applications and micro services at very less time; - No XML based configurations. Very much simplified properties. The beans are initialized, configured and wired automatically; - The Spring Boot artifacts can be deployed directly into Docker containers.	- Lots of JAR files and settings are created, which unnecessarily increase the deployment binary size with unused dependencies as well as impact the performance.

6. ASP.NET with NuGet

Pros	Cons
- NET is very good at "write once, use often"; - Performance is excellent; - Security is solid; - Supports from Microsoft.	- Older versions of ASP.NET have issues needing to be upgraded; - There are version conflicts in older versions; - Third party integration can sometimes take extra work. - Not great in cross-platform support

7. Express (Node.js/JavaScript) with Mocha Test

Pros	Cons

- I/O request handling: Node.js together with Express.js is capable of supporting thousands of concurrent actions; - Fast app development: allows you to use the same language which is JavaScript both on the back-end and front-end; - Easy integration of third-party services and middleware; - Easy to learn.	- Not concise in coding: nested callback functions directly impact the quality of code; - Immaturity of tooling: registry is not structured well enough to offer the tools based on their rating or quality.

Part 3 – Continuous Integration

1. CircleCI

Pros	Cons
Easy and fast to start; Free plan is provided for enterprise projects; Lightweight and easily readable YAML configurations; Does not require any dedicated server to run CircleCI; Support a variety of languages: Go (Golang), Haskell, Java, PHP, Python, Ruby/Rails, Scala.	CircleCI supports only two versions of Ubuntu for free and MacOS as a paid part; Third party software may be required to make customizations; Being a cloud-based system, it has the likelihood to stop supporting any software.

2. Travis CI

Pros	Cons
Build matrix out of the box; Fast start; Lightweight YAML configurations; Free plan for open-sourced projects; No dedicated server required;	Price is higher compared to CircleCI, no free enterprise plan; Customization often requires third party softwares;

3. Jenkins CI

Pros	Cons
Economical since it is free; It is a plugins system which means highly compatibl; It gives full control of the system.	Dedicated server (or several servers) are required, which means additional expenses. Time needed for configuration / customization

Part 4 – Database and Cloud Storage

1. MongoDB + NoSQL

Pros	Cons
- Free: open source project and is completely free; - Flexibility: there are lot of extensions and libraries, which makes life easier. Eg. Mongoose	- Sometimes it is not easy to find support for a NoSQL Database. - Schema-less data storage, data redundancy, non-ACID.

2. AWS Aurora + MySQL

Pros	Cons
- High Performance and low latency - Security: top in the industry; - Compatible with MySQL and PostgreSQL: provide convenience to a team with different background; - Great Scalability.	- Powerful and able to handle high traffic sites; - Feature rich; - Many security features built-in; - User management capabilities.

3. SQLite

Pros	Cons
- Serverless which means it is simple to set up and zero configuration is required; - File-based system makes it very portable; - Great for development and testing.	- Doesn't provide network access (i.e. accessing it from another machine) as it is serverless; - Not built for large-scale applications; - No user management.

4. SQLAlchemy

Pros	Cons
- Support Python development environment well; - Has thorough documentation and it is actively maintained; - Has great ORM for manipulating database without writing SQL queries.	

Tech Stack

- Our setup:
- Frontend: React, React-bootstrap
- Backend: NodeJS,
- Database: MongoDB
- CICD: Circle CI, Heroku