Question 1

a.

The backend framework I'd prefer to use is Spring Boot. The reasons for using Spring Boot for this particular web application are:

1. **Autoconfiguration:** Spring applications require dependency configuration for its working. Spring Boot automatically configure the pre-set of dependencies required; there's no need for manual configuration. This greatly increases the efficiency and reduces errors in the developmental process.
2. **Multithreaded**: The website given requires a large amount of computing and processing. In order to meet the particular performance requirements, Spring Boot's multithreaded feature can come into play. As Spring Boot is multithreaded, it can handle multiple processes at a time.
3. **Opinionated Approach:** Spring Boot ensures a high degree of verbosity in the configuration files. This aids in viewing the configuration process in a seamless manner. It also does all the heavy lifting; in that way, the developers can pay attention to the actual problem at hand. Spring Boot autoconfigures a set of default beans and always chooses the more sensible option depending on the constraints. Spring Boot being opiniated in nature makes it more dynamic and more responsive to change.

b.

The frontend framework I would prefer to use for this particular application would be React.js. The reasons for using React.js for this particular web application are:

1. **Reusable UI Components**: There are a number of reusable UI Components within the website such as the dropdown menus in the navigation bar (Explore, Types, Services, Research); the Latest Reviews, Latest How-to and the Latest on the Water section; and so on. React.js supports creating specific UI interfaces from components (JavaScript Functions). Rendering them into the framework is easy as it can be called directly as an HTML element. Using reusable components makes the website easier to maintain and simpler to change if required. Furthermore, the components can be nested into other components to make more complex

components. For example, the filter at the left of the webpage can be considered as one component. And the various dropdown boxes can be considered as the same component repeated. In the webpage, we can nest the dropdown box components into the filter component.

2. **Virtual DOM**: Websites built with HTML, CSS and vanilla JavaScript are required to modify the DOM immediately if a change in the UI is required. This continuous modifcation of the DOM is costly and inefficient. Even if a miniscule portion of the UI is changed like fixing a spelling error, the entire DOM needs to be updated and rendered again. However, React.js makes use of a Virtual DOM. Whatever updates to the UI are first committed onto the Virtual DOM. The ‘most recent’ Virtual DOM is then compared with the previous ‘version’ of the Virtual DOM. (This comparison process is known as **diffing**). Whatever modifications are made will then be updated in the actual DOM. Essentially, the use of the Virtual DOM in React.js allows for specific modification and updating, which in turn leads to better performance and efficiency.
3. **Built-In Hooks and Testing**: Hooks are functions that allows using React features within the components. The usage of hooks allows for increased reusability of code and efficient organization of code. Hooks also ease the testing and quality analysis portion of development as they extract the stateful logic. Testing libraries such as Jest and Enzyme can be used to test the hooks specifically.
4. **Built-In Libraries:** React.js is fitted and compatible with a variety of libraries and packages such as React-Bootstrap, Material UI, etc. for the design portion as well as Redux, React Router, etc. for managing the overall application states.

C.

The database I would prefer to use for this web application is MongoDB. The reasons for choosing MongoDB are as follows:

1. **NoSQL:** NoSQL is perfect for applications and software that require multiple iterative modifications. AS boats.com needs to constantly be updated in terms of new sales, deals, boat models, etc. it’s imperative for it to be reactive and dynamic. In contrast, if any SQL based database were to be used, it would be very costly and

inefficient to make changes and modifications as SQL based databases work well with applications with minimal modification.

2. **Horizontally Scalable:** As MongoDB isn't a relational database, it can be scaled horizontally, which can be done simply by adding more servers. In relational databases however, the database can only be scaled vertically, which basically means it can only be scaled up to a certain limit before it needs to be replaced completely.
3. **Security:** MongoDB offers a variety of security services such as Authentication, Auditing, Encryption, and so on. This particular feature is crucial for a web application such as boats.com as the website handles multiple and large monetary transactions.
4. **Indexing:** As MongoDB is document based, it's fitted with indexing to access the document elements. This allows the particular web application to increase performance and efficiency and overall deliver the result to the customer with minimal delay.

d.

For version control of the codebase, I would make use of Git (Global Information Tracker) for the following reasons:

1. **Security:** Git has a number of security features and also uses a cryptographic algorithm known as SHA1 in order to safely encrypt the data and files to prevent accidental and malicious modification.
2. **Performance:** Git is very performance oriented due to the presence of advanced and complex algorithms, branching and merging capabilities, and so on.
3. **Distributed Development**: Like Darcs, BitKeeper, Mercurial, Bazaar, and Monotone, Git gives each developer a local copy of the entire history of the development and whatever modifications made are copied from one such repository to another. These changes are imported as added development branches and can be merged in the same way as a locally developed branch.
4. **Garbage Accumulation:** Aborting operations or backing out changes will leave useless dangling objects in the database. These are generally a small fraction of the continuously growing history of wanted objects. Git will automatically perform

garbage collection when enough loose objects have been created in the repository. It can be called explicitly using git gc.

e.

The platform I would prefer to use for media storage is Cloud Based Storage Services, specifically Amazon S3. The reasons are as follows:

1. S3 has virtually unlimited storage. The limit for a single file in S3 is 5TB, which is a massive amount of data.
2. S3's designed with 11 9's of durability, which ensures media safety and protection
3. S3 has multiple storage classes that can be modified based off of the application's requirements. Examples include S3 Intelligent-Tiering, S3 Standard, and so on.
4. The data stored is easy to fetch and retrieve due to S3's cross region replication feature

f.

I would deploy this web application on a cloud platform, specifically AWS and I would utilize Amazon Elastic Beanstalk for the following reasons:

1. It quickly scales the web application and immediately gets it up and running which is required for a web application like boats.com
2. It's managed by AWS and it automatically handles a barrage of tasks such as deployment, load balancing, health monitoring, and so on.
3. There is technically no additional charge with Elastic Beanstalk; the pricing will be applied if Elastic Beanstalk makes use of paid services
4. Elastic Beanstalk works in tandem with Amazon S3, which is useful for media storage interaction
5. Elastic Beanstalk is Platform as a Service (PaaS), which provides users with a preset server for use. But, if users need more customization, Elastic Beanstalk can be modified to only act as a vessel for the web deployment.