Description of System

For an application with 100 users, we can use a traditional web application design. The app is architected into three tiers:

- Presentation tier - through the frontend encompassing a User Interface.
- Application tier - consisting of a Web Server at the backend.
- Data tier - comprising Databases and File System, also backend.

Starting with the backend, we are using PostgreSQL databases to hold Live and Historical game data. The Live database will hold real-time stat lines of games that get filled in more as the games progress while the Historical database will hold data from all games played in the past. Both databases are replicated for backup and disaster recovery.

The web application, hosted on the web server, will communicate with the databases. The application will retrieve data from the Historical database for the User Interface, and will perform CRUD functions on the Live Database. This web server also connects to a File System which contains the HTML design for the UI and images to complement the visual presentation. All of these listed components form the backend of the application.

Moving to the frontend, the UI, which runs on HTML and Javascript, is accessed by coaches, statisticians, and the review committee. Coaches will request data from both databases via the UI. The Live Basketball Statistician records data from ongoing games and sends it to be stored in the live PostgreSQL database. Meanwhile, the Review Committee requests data from the Live Database, ensures its accuracy, then sends it to the Historical Database.

Technology Used: To ensure flexibility, scalability, cost effectiveness, and ease of access, this application can be hosted on the cloud, such as Amazon Web Services (AWS). In using the cloud to store our PostgreSQL databases and our web server, we have three options:

- Host the databases and the web server on the same EC2 instance.
- Host the databases on one EC2 instance and the web server on a serpetate one.
- Host the databases by using Amazon Relational Database Service (RDS) and host the web server on an EC2 instance.

Hosting it on the cloud allows for a pay-as-you-go model. Due to the low user count, web traffic will be inconsistent so ensuring our costs match the processing power needed is very crucial.

Issues caused by 10,000 users: Our design supports a low number of users. If we were to expand the user base to 10,000, the system would experience the following issues:

- System performance and task completion would slow down due to bottlenecks being formed from excessive requests on the web server.
- Data retrieval and updates would slow thus impacting real-time data access.
- User experience would be poor. For instance, coaches needing stats during a game, especially during a short timeout, would find the delayed service frustrating.
- Error and issue tracking would become unmanageable, making it difficult to take action in a timely manner.