DESIGN DECISIONS

1. **Database Storage Decision**: To prevent potential overflow issues, I decided to store Fibonacci values in the database as strings rather than numerical data types. This approach ensures that large Fibonacci values are accurately represented and avoids truncation or loss of precision during storage.

2. **Precomputation and Caching Strategy**: a. To optimize the application's performance, a precomputation strategy was implemented. When a user requests a specific range of Fibonacci values, such as the first 5, the application precomputes and stores these values in the database. Subsequent requests for the same range can be served directly from the stored data, reducing computation time.

b. To further enhance the efficiency of Fibonacci value retrieval, a caching mechanism was employed. The application maintains a state that includes the last computed index, the corresponding Fibonacci value, and the value of the Fibonacci sequence at index i-1. This information allows the application to avoid redundant computations when a user requests values beyond the last computed index. For instance, if the last computed Fibonacci index was 10 and the user requests the Fibonacci value at index 20, the application can efficiently compute values starting from index 11, utilizing the concept of caching.

By adopting these design decisions, the application aims to optimize computation and storage efficiency, ensuring accurate representation of large Fibonacci values and delivering fast responses to user requests for Fibonacci sequences.