To run the application:

- Ensure that ScyllaDB is installed and running.
- Set up the todos_keyspace keyspace in ScyllaDB.
- Install the required dependencies using go mod tidy.
- Run the application using go run main.go.

Explanation of the design decisions made during development:

Use of Golang: Golang was chosen as the programming language due to its simplicity, efficiency, and strong standard library. It's well-suited for building high-performance web servers and handling concurrent requests.

Use of Gorilla Mux: Gorilla Mux is a powerful URL router and dispatcher for matching incoming HTTP requests to their respective handler functions. It was chosen for its flexibility and ease of use in defining RESTful API routes.

Database Choice - ScyllaDB: ScyllaDB, a highly available distributed NoSQL database compatible with Apache Cassandra, was chosen for its scalability and fault tolerance. It's suitable for handling large volumes of data and high throughput.

CRUD Operations: The API implements CRUD (Create, Read, Update, Delete) operations for managing todo items. Each operation corresponds to an HTTP method (GET, POST, PUT, DELETE) and a specific endpoint.

UUID Generation: To ensure the uniqueness of todo IDs, gocql's TimeUUID() function is used to generate UUIDs based on the current time. This helps in avoiding conflicts when creating new todo items.

Pagination: Pagination support is included in the ListTodos handler to retrieve a subset of todos based on the requested page and page size. This helps in managing large datasets efficiently.

Error Handling: Basic error handling is implemented to handle invalid requests, database connection errors, and other potential issues gracefully. Error responses with appropriate HTTP status codes are returned to the client for better error diagnosis.

Timestamps: Created and Updated timestamps are included in the Todo struct to track when each todo item was created and last updated. This information can be useful for auditing and tracking changes over time.

HTTP Server: The API is served using Go's built-in HTTP server. It listens on port 8080 by default and routes incoming requests to the appropriate handler functions using Gorilla Mux.

Modular Structure: The code is organized into separate files (database.go, handlers.go, main.go) for better maintainability and readability. Each file contains related functionality, making it easier to understand and modify individual components.