AWS 3-Tier Architecture Overview

This architecture is implemented for the web application and consists of three distinct layers: the Presentation Layer, the Application Layer, and the Data Layer. Each layer is responsible for a specific part of the application's functionality and is hosted using specific AWS services.

1. Presentation Layer

- **AWS Services**: Amazon CloudFront and Amazon S3
- **Purpose**:
    - **CloudFront**: Acts as a Content Delivery Network (CDN), distributing static content such as HTML, CSS, and JavaScript files globally. This setup reduces latency for users around the world by caching content at edge locations, ensuring faster access to the application.
    - **S3**: Simple Storage Service (S3) is used to store static files. In this case, it hosts the React.js frontend of the application, with the files being distributed via CloudFront to ensure optimal performance.

2. Application Layer

- **AWS Services**: EC2 (Elastic Compute Cloud) or Elastic Beanstalk
- **Purpose**:
    - This layer hosts the Django application backend. Two options are utilized:
        - **EC2**: Provides virtual servers where Django can be installed and configured manually.
        - **Elastic Beanstalk**: A platform-as-a-service that simplifies the deployment process, handling load balancing, scaling, and monitoring of the Django application, allowing more focus on code development.

3. Data Layer

- **AWS Service**: Amazon RDS (Relational Database Service)
- **Purpose**:
    - Amazon RDS manages the database, which stores tasks and other persistent data. It supports various database engines, including MySQL and PostgreSQL. This service handles backups, patching, scaling, and replication, making it easier to manage and scale the database effectively.

How These Layers Work Together

1. **User Requests**:
    - When a user accesses the web application, their request first reaches **CloudFront**, which checks if the requested files (React app) are cached. If cached, the files are served directly from the nearest edge location to improve speed.
2. **Serving Static Content**:

    - If the requested content is not cached in CloudFront, it fetches the files from **S3** and then serves them to the user. This allows the user to view the frontend of the application.
3. **Backend Requests**:
    - When a user interacts with the application (e.g., adding a task), the frontend sends requests to the Django backend, which is hosted on **EC2** or **Elastic Beanstalk**.
4. **Database Interaction**:
    - The Django application processes the request and interacts with the **RDS** database to store or retrieve data, such as adding or fetching tasks.
5. **Response to User**:
    - The processed data is sent back through the application layer and presented to the user via the frontend.

Summary

- **CloudFront + S3**: Serve the static frontend (React.js).
- **EC2/Elastic Beanstalk**: Host the dynamic backend (Django).
- **RDS**: Manage the database for persistent storage.