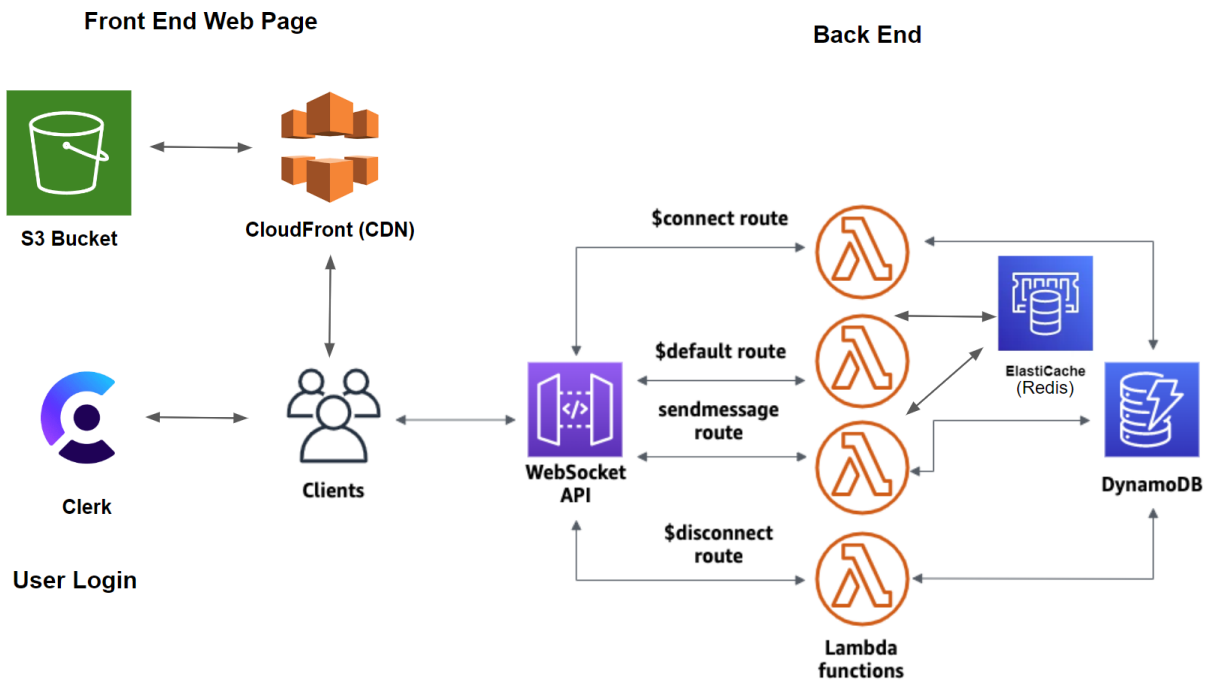
CSC409 Group88 Assignment 3 Report

Feiran (Philip) Huang

Jingwen (Steven) Shi

Xuankui Zhu

System Design



System Diagram

The r/place frontend, residing in the AWS S3 bucket as a static website distributed by CloudFront as a CDN, serves as the interface users interact with. When a user interacts with the frontend, it initiates a WebSocket connection with AWS API Gateway through the WebSocket API. This WebSocket connection facilitates the transmission of various commands, each triggering specific actions involving AWS Lambda functions, and the corresponding Lambda function retrieves data from the correct database or cache. The user is asked to log in using Clerk (a third-party authentication service) before drawing on the canvas to prevent abusing our service.

- **Connect Route**: This route initializes a connection and logs the user's connection state, marking their presence within the system.
- **Default Route**: On this route, a request retrieves the entire canvas from Redis, a data store. Redis holds the current state of the canvas, which is returned to the user.
- **Sendmessage Route**: When a user draws something, this route is triggered. It first invokes a GET request for the timestamp of the user's latest drawing, and if the difference between the current time and the retrieved timestamp is longer than 5 minutes, it records the user's drawing in DynamoDB and Redis, updating both databases with the new information. Afterward, it broadcasts this update to other connected users, ensuring everyone sees the latest drawing.
- **Disconnect Route**: Finally, the disconnect route handles the termination of a user's connection. It removes their connection state from the system, signalling their departure.

This setup orchestrates a smooth exchange between the frontend and backend: WebSocket APIs manage real-time communication, Lambda functions execute specific actions based on received commands, while DynamoDB and Redis store and distribute data, ensuring a collaborative and responsive experience within r/place.

Services are deployed using CloudFormation, enabling us to efficiently modify, reuse, and maintain version control over our systems. They're monitored through CloudWatch, empowering us to observe, analyze, and manage resources and applications effectively and actively.

Strength

1. **Real-time interaction**: Using WebSocket enables real-time bidirectional communication, creating a highly interactive user experience. Users can see updates in near real-time, enhancing engagement.
2. **Scalability**: Lambda functions are scaling automatically. They execute on-demand, which ensures efficient resource utilization.
3. **Easy to deploy**: All the services are serverless, so developers do not need to config and setup any server and environment to run these services
4. **Easy maintenance**: Different component handles specific tasks. This modular approach leads to better maintainability, scalability, and ease of updates.
5. **Partition tolerance**: Using Redis as cache and DynomoDB as database, the system functions regularly even when one of them is down.
6. **Availability**: The system will always return the correct board data. Redis is deployed in two different available zones with one primary and one replication.

Weakness

1. **Complex in debugging**: Serverless applications may not always provide comprehensive error outputs, sometimes lacking sufficient information. This limitation can hinder effective debugging processes, making diagnosing issues more challenging.
2. **Cost Uncertainty**: Serverless computing is charged based on usage when a high workload and unexpected access can lead to unintended costs.
3. **Consistency**: If users drawing messages lost during sending to the backend, the draw would stay at the frontend but not broadcast to other users, which causes an inconsistency. Also writing into Redis and Dynomo is not simultaneous, which causes inconsistency.
4. **Security**: All webpages and codes are stored in S3 Buckets, which are prone to attacks.

List of Services

- **S3 Bucket**
  - S3 Bucket hosts and serves webpages (html, css js) as frontend server
- **API Gateway (WebSocket)**
  - Sends user action data to a websocket and invokes Lambda functions
- **Lambda**
  - Each function handles a specific functionality, including connecting and disconnecting users, retrieving the board, and delivering user messages
- **DynomoDB**
  - Stores the state of the board, and also information of users such as the time of a user's latest edit
- **Elastic Cache (Redis)**
  - Stores the state of the board, allows users to retrieve data
- **CloudFormation**
  - Configures all services within the architecture and starts them at once
- **IAM**
  - Manages user permissions
- **VPC, NAT, Security Group, Subnet**
  - VPC is used to achieve private cloud isolation from the public.
  - The security group is used for controlling inbound and outbound rules.
  - NAT and subnet are used to allow communication between services within the VPC and the outside world.