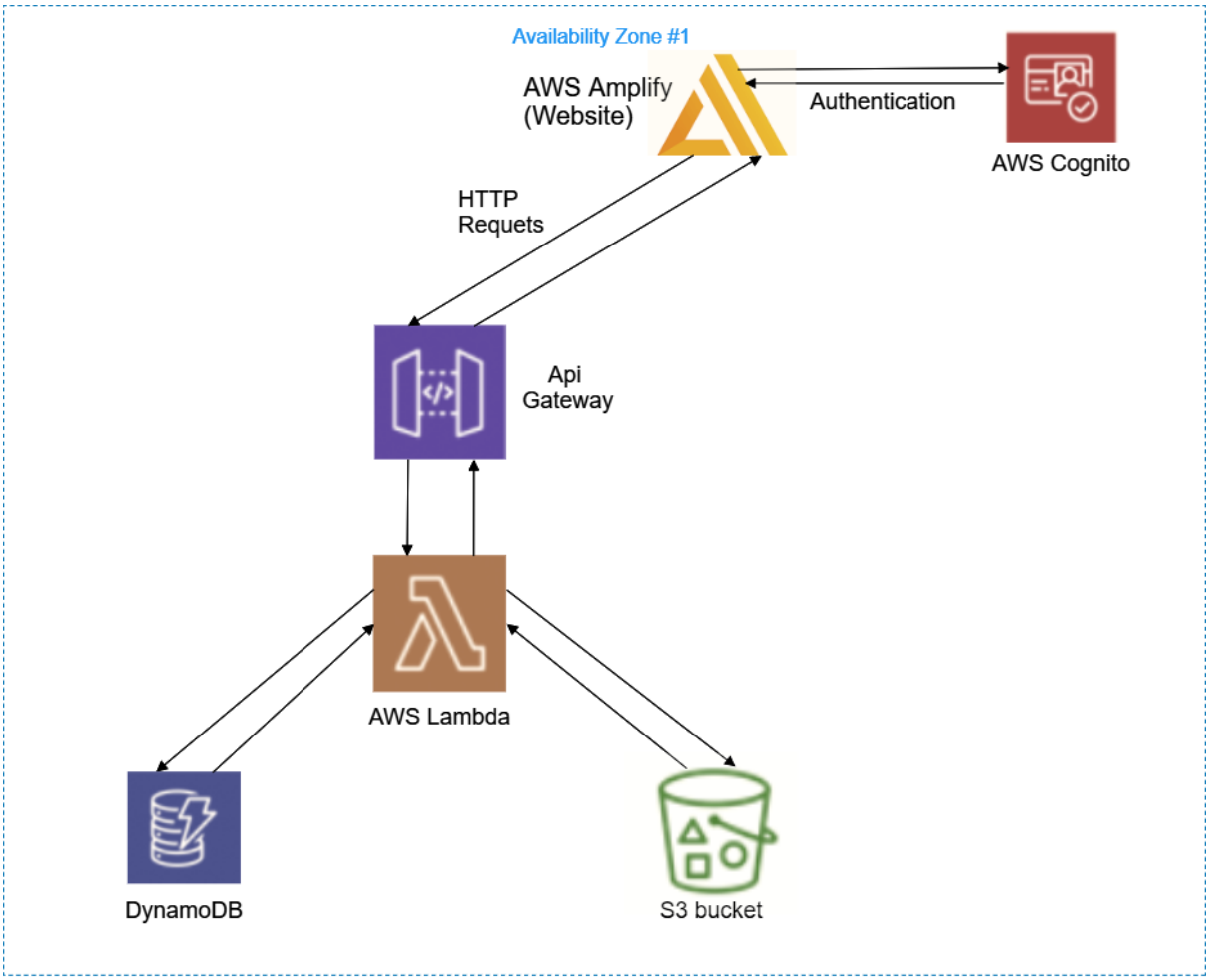
Cloud Computing - project presentation

Paulina Pacyna

Mateusz Wójcik

Mateusz Marzec

Diagram



Functional requirements

- Each person can register an account and fill his profile with information (and edit, information stored in DynamoDB and files - possibly image and recording stored in S3)
- People not logged in cannot search and view other profiles
- Authentication page is the root page
- Authentication is implemented using AWS Cognito

- Each person can search other profiles on the main page, fetched from DynamoDB using Lambda and Api Gateway
- Searching by name and surname.
- The response from the RDS contains people that match the search string and contains only 15 first results to avoid network congestion.
- Profile details are visible on a separate site and contains personal information along with the recording
- Profile details can be provided at any time, not only during registration.
- The user can change profile details.
- Recording can be reuploaded and deleted.

Nonfunctional requirements

- Profile details and the recording are durable.
- User login data is encrypted on the rest and the move.
- Data integrity provided by DynamoDB and S3 services.
- Scalability is obtained by AWS Lambda. Lambda automatically scales to handle 1,000 concurrent executions per Region.
- High availability will be provided using Route 53 by placing API Gateways in different Availability Zones. The traffic in one region will be redirected to the nearest API Gateway.
- The data of users will be backed up using DynamoDB point-in-time recovery and S3 cross-region replication.

Resources

- **DynamoDB** - for storing data about each user (in case of mp3 and jpg files it will be link to the S3 bucket)
- **Cognito** - user authentication provided for creation of the profile. User pool employed in Cognito consists of email address and password chosen by the end user. Users' accounts are verified using email authentication providing the link to activate the account.
- **Lambda and Api Gateway** - two REST endpoints - one for getting and posting to DynamoDB - second for uploading to S3
- **Amplify** - used for deploying frontend web application made in React
- **S3** - storing mp3 (recordings of the end user) and image (profile pictures) files

Backup and Disaster Recovery plans

- We plan to replicate our data into another region and provision a copy of our core infrastructure. Resources required to support data replication and backup such as DynamoDB and S3 are always on. Other elements such as application servers are loaded with application code and configurations, but are switched off. We plan to

update our data in another region every 24 hours (AWS Data pipeline, S3 cross region replication). Health checks can be made for example with CloudWatch.

- Already provided backup of DynamoDB database. We also plan to enable point-in-time recovery that will enable restoring the database from each point in time for the last 35 days.
- Finished application will also employ S3 cross-region replication in case of disasters in a particular region. Also the size of the bucket will increase and it will be difficult to store the backups on local machines. We also want to provide a life cycle in order to store and archive older files.
- Lambda is highly available as it runs the function in multiple availability zones by default
- We plan to place API gateways in two AZ and connect them using Route 53 (using edge-optimized API endpoints), to obtain high availability. The traffic in one region will be redirected to the nearest API Gateway.

RPO: up to 24 hours

RTO: up to few hours