Functional requirements

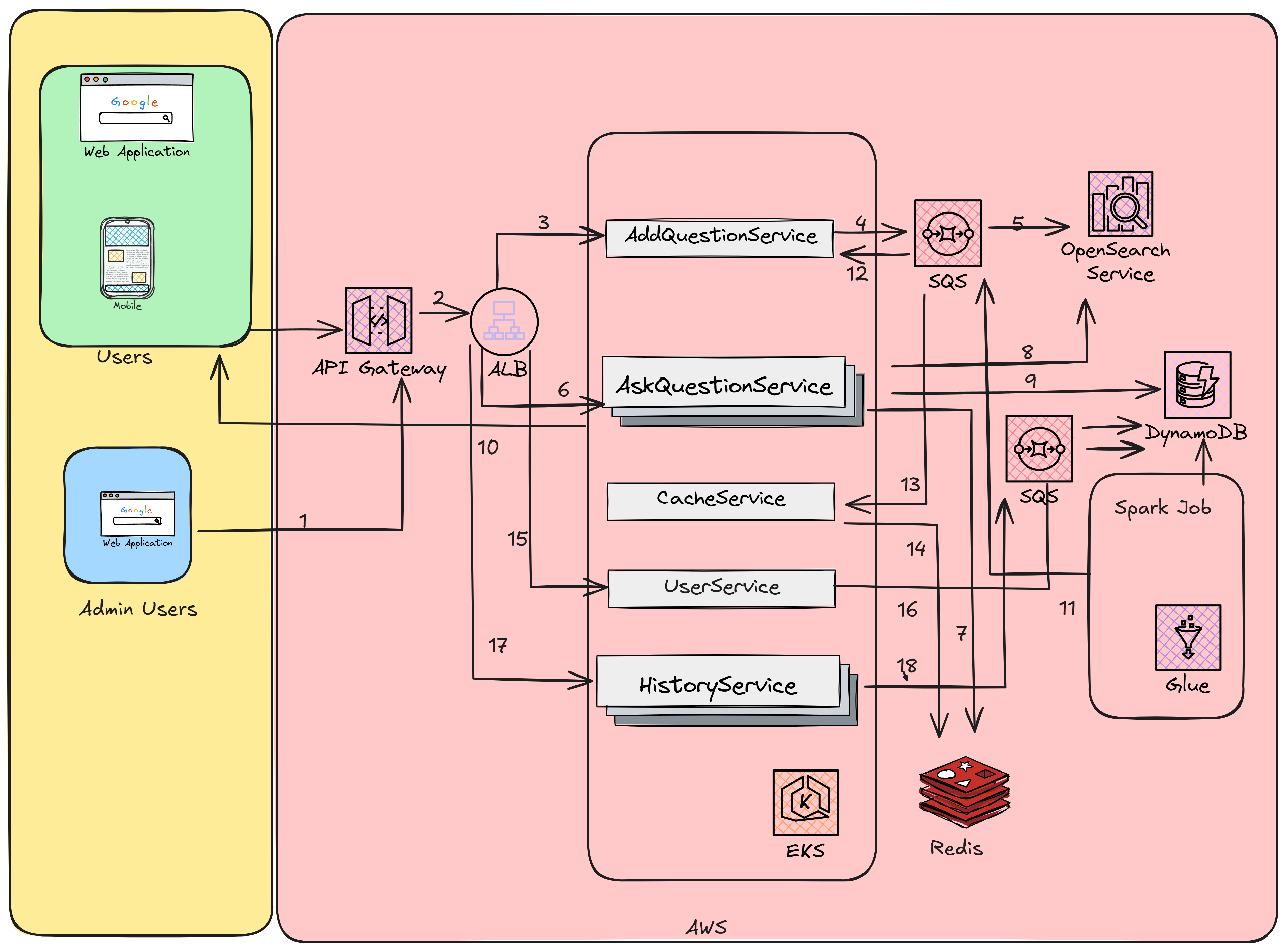
* Respond to predefined basic questions (e.g., "What is your name?").
* Accurately evaluate and return the result of arithmetic expressions.
* Engage with users through simple text-based queries.
* Store predefined questions and answers.
* Persist user query history for future retrieval.
* Support creation of user accounts

Non-Functional Requirements

* The system should be highly scalable. Expected Traffic can go up or down any time
* System should be Highly available.

Resource Estimation

* Traffic Estimation: 100 QPS (at peak, read Predefined queries), 100 QPS (write for user logs).
* Storage Requirements
* Predefined Questions - 100 questions/day, 100KB each → ~10MB/day, ~300MB/month ~4GB/year.
* User logs - 200KB per query, 150 million queries → ~30TB/year.
* No of Servers
* 20 instances of app server with Autoscaling
* 10 node cluster for storing predefined questions with Autoscaling
* DB provisioned with autoscaling enabled to handle 100 QPS write load.
* High Level Design



Core Components

* **API Gateway** - The AWS API Gateway is the managed service which maintain and manage the web-services. It will provide features like Authorization, Rate limiting, monitoring and logging
* **EKS** - The services will be deployed in Elastic Kubernetes service which is highly scalable to manage the high volume of queries from users. It is available across multi-AZ, supports auto scaling of pods and containers, and cost effective
* **ALB** – The Application Load Balancer is used in conjunction with EKS to manage networking to provide a stable endpoint for the pods that the API Gateway can route to.
* **OpenSearch** – is another managed service of AWS which will store the predefined questions, it can scale automatically handling ensures optimal performance during peak usage and cost efficiency during low-traffic periods. It also supports full text search and near real time search.
* **SQS** – Simple Queue service is used to provide decoupling between microservice, analytics and DB layer, It provide seamless integration with other amazon services like DynamoDB and OpenSearch, is highly scalable and can support high throughput.
* **DynamoDB** – DynamoDB is highly scalable, highly write optimized managed DB provided by AWS. It can scale to millions of write per second and also cost effective. Also, it integrates seamlessly with AWS services like SQS.
* **Glue** – is the serverless and managed scalable data integration service which will run the analytical spark jobs in a scheduled manner.
* **Redis –** Redis is the managed cache service which is highly scalable and integrates seamlessly with other AWS service like SQs etc.
* API Design
  + POST /ask

Request: {“user\_id": "123", "query": "2+2", "type": "ARITHMETIC", "askedAt": "2024-01-01T12:00:00Z" }

Response: {“answer": "4" }

* + POST /add-question

Request: { "id": "q1", "question": "What is your name?", "answer": "RoboBob", "type": "BASIC"}

* + GET /history/{user\_id}

Response: [ { "id": "q1", "query": "2+2", "answer": "4", "type": "ARITHMETIC", "askedAt": "2024-01-01T12:00:00Z" }, { "id": "q2", "query": "10\*5", "answer": "50", "type": "ARITHMETIC", "askedAt": "2024-01-01T12:05:00Z" } ]

* + POST /register

Request: { "username": "john\_doe", "password": "\*\*\*\*" }

* DB Design
  + **predefined\_questions** (id [String], question [String], answer [String], type [String], addedAt [String], addedby[String] )indexed in OpenSearch
  + **query\_history** (id [String], user\_id [String], question [String], answer [String], type [String], askedAt [String])
  + **users** (user\_id [String], username [String], password\_hash [String], email [String], created\_at [String])
* Detailed Design
* The pre-defined questions will be stored in indexes in OpenSearch to ensure a response time in milliseconds while handling millions of QPS. In the indexes the questions will be used keyword which will ensure an exact matching without tokenization, thus resulting a response time in millisecond

PUT /questions\_index { "mappings": { "properties": { "question": { "type": "keyword" }, "answer": { "type": "text" } } } }

* The admin users put the predefined questions in the OpenSearch indexes through the AddQuestionService deployed as pods in the EKS
* The users sends the basic questions or arithmetic expression via the API Gateway to the AskQuestionService, deployed as pods in EKS, based on the type of the questions either evaluates the arithmetic expression, or searches for the basic question, at the same time it pushes the query with other details like timestamp, type etc to the SQS queue to be stored in the DynamoDB table for user historical data.
* The Spark Job executed via Glue as a batch job on predefined frequency does the analysis on the historical data for the users and analyses insights like which are the most searched queries, what queries were not answered, and sends them via an SQS queue to AddQuestionService for further action and CacheService to put the most frequently accessed queries in cache.
* The HistoryService gets the historical data for the user from the DynamoDB table and helps presenting in a user friendly manner to the users