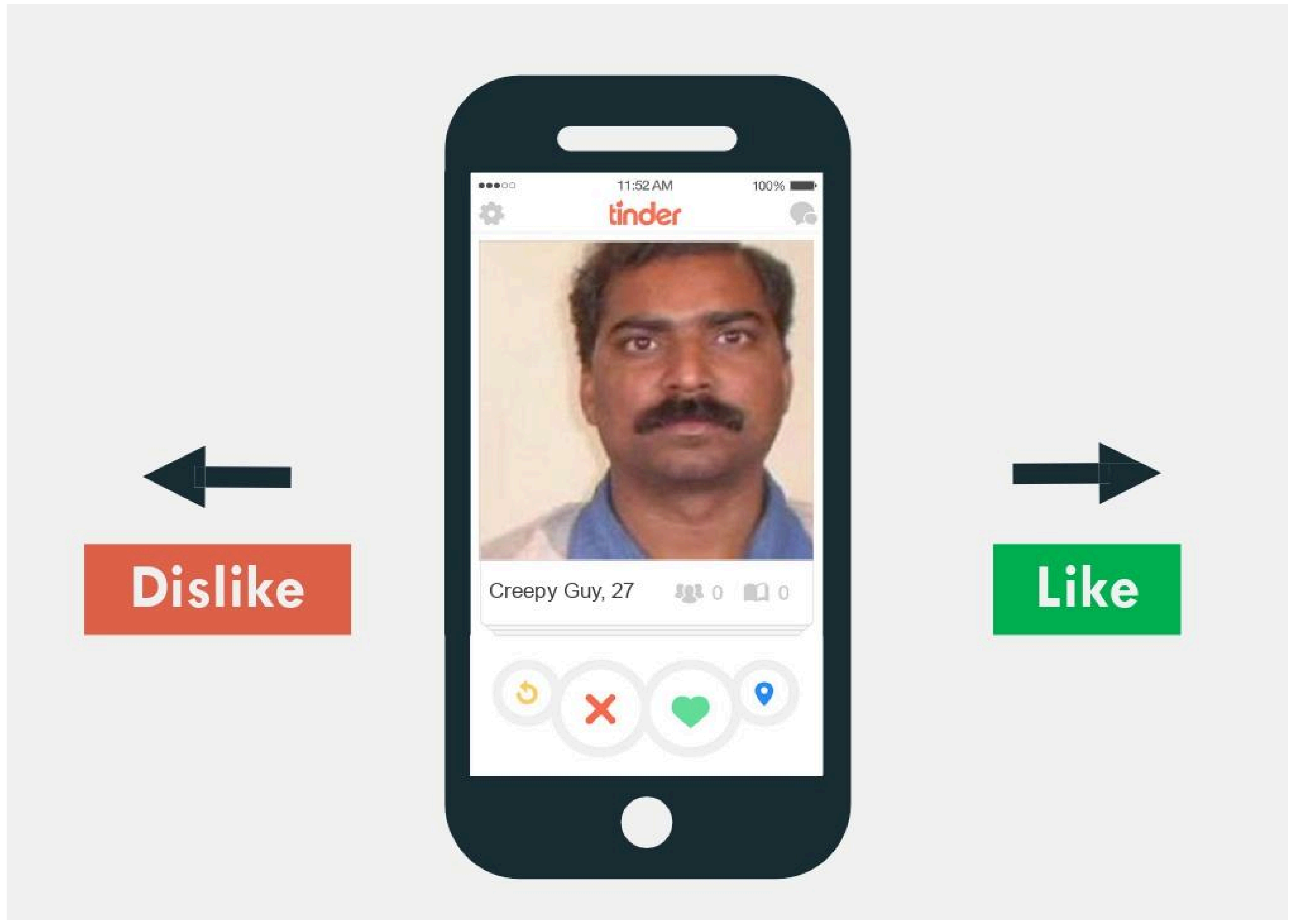
tinder
Case Study

What needs to be done?

Registration
Retrieving potential matches
Swipe on users
Match with users

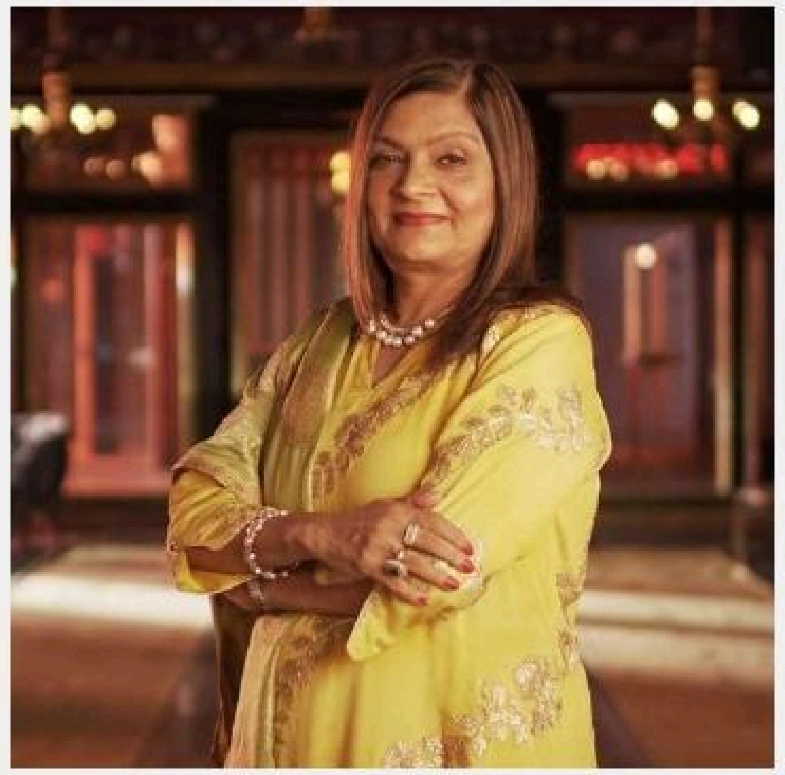
Understand the system supporting these features and design the API & data sent

Understanding the platform



Key Points

- 50 million users
- Won't focus on super-liking, or undoing swipes
- Won't focus on matching algorithm - handled by different teams:
  - ML algorithms
  - Rishta aunties – Seema Aunty



Gather Requirements

100 TB	5 TB
Profile Photo Storage	Metadata Storage
Google Cloud Storage	PostgreSQL

users

Stores all of the users

userID	varchar(30)	PK
firstName	varchar(30)	
lastName	varchar(30)	
email	varchar(30)	
password	varchar(30)	
geolocation	points	
dateOfBirth	date	
photosURL	List<string>	

potential_matches

Stores all of the potential matches that a user can swipe through

pmID	varchar (30)	PK
user1ID	varchar (30)	FK users (userID)
user2ID	varchar (30)	FK users (userID)

swipes

Stores all of the swipes that a user has made

swipeID	varchar(30)	PK
swiperID	varchar(30)	FK users(userID)
swipeeID	varchar(30)	FK users(userID)
swipeType	varchar(10)	
timestamp	timestamp	

matches

Stores all of the matches (both users have swiped right)

matchID	varchar(30)	PK
user1ID	varchar(30)	FK users(userID)
user2ID	varchar(30)	FK users(userID)
timestamp	timestamp	

users

Stores all of the users

userID	firstName	lastName	email	password	geolocation	dateOfBirth	photosURL
z1234	Shrenik	Surana	ss@gmail.com	e9fa9wf89au32r	43.644899, -79.383905	Jan 1, 1989	https://cloud.google.com/w2i39fjaef.jpg https://cloud.google.com/43t3etw4er.jpg https://cloud.google.com/k67u45y4rr.jpg
23r8w	Priyanka	Chopra	pc@gmail.com	adfaweafawefa	67.644899, -107.383905	Jan 1, 1986	https://cloud.google.com/34rfaewf.jpg https://cloud.google.com/sg5hdty45.jpg https://cloud.google.com/rth45ytrt.jpg

potential_matches

- → Stores all of the potential matches that a user can swipe through
- → Remove potential match once user is swiped on (either left or right)
- → Updated every day, unless the user isn't active
- → Populated by a different team using ML algorithms

pmID	user1ID	user2ID
pm1234	z1234	23r8w
pm2345	z1234	92345
pm3456	z1234	12934
pm4567	z1234	38452
pm5678	z1234	hjf8a9

Other Tables

- dating_preferences
- user_preferences

swipes

- Stores all of the swipes that a user has made
- Updates once a user has swiped either left or right
- swipeType will be either "like" or "pass"

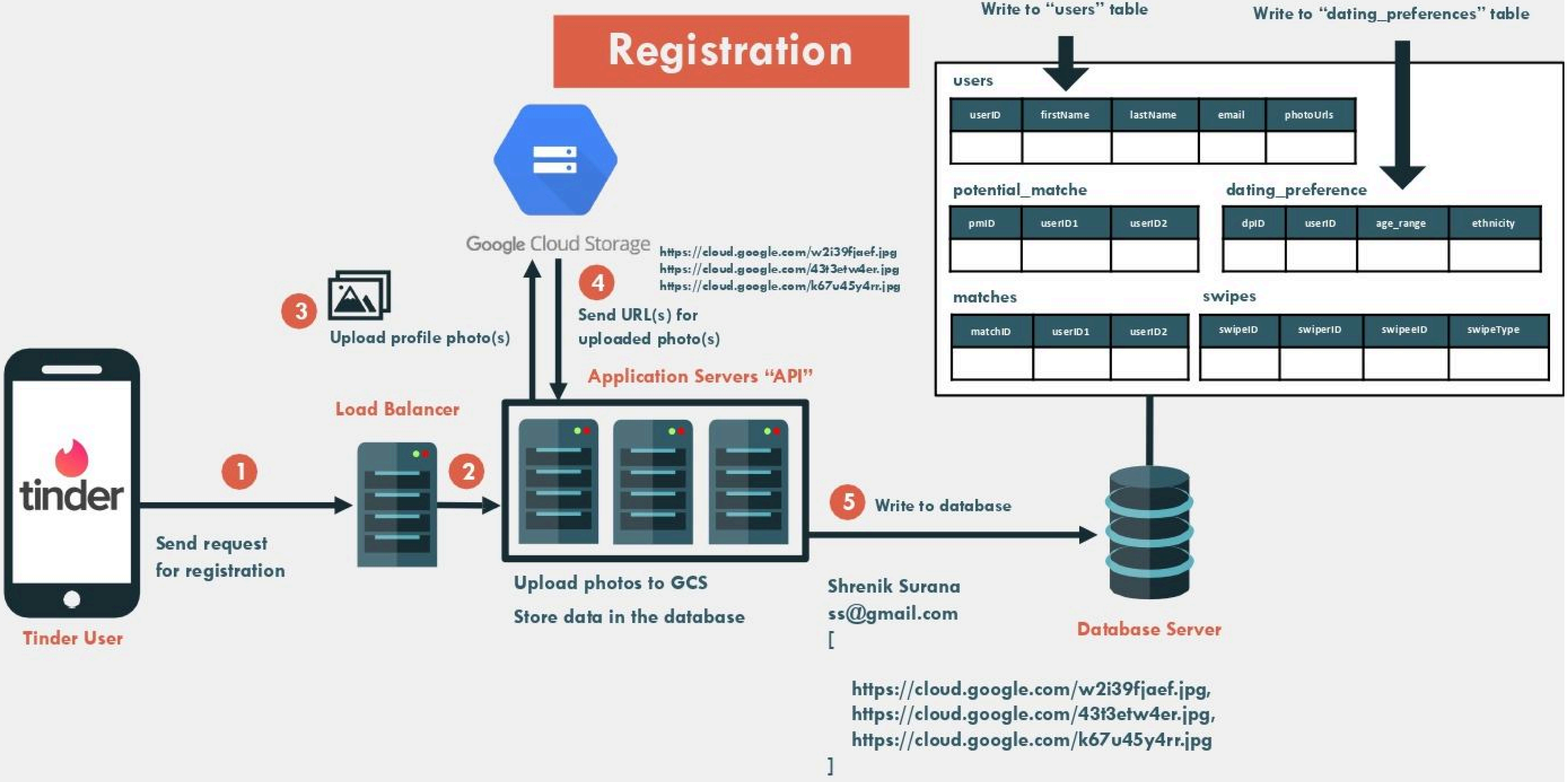
swipeID	swiperID	swipeeID	swipeType	timestamp
s1234	z1234	23r8w	like	2023-07-30 13:00:00
s3456	z1234	92345	pass	2023-07-30 13:00:06
s489d	23r8w	z1234	like	2023-07-30 14:00:15

matches

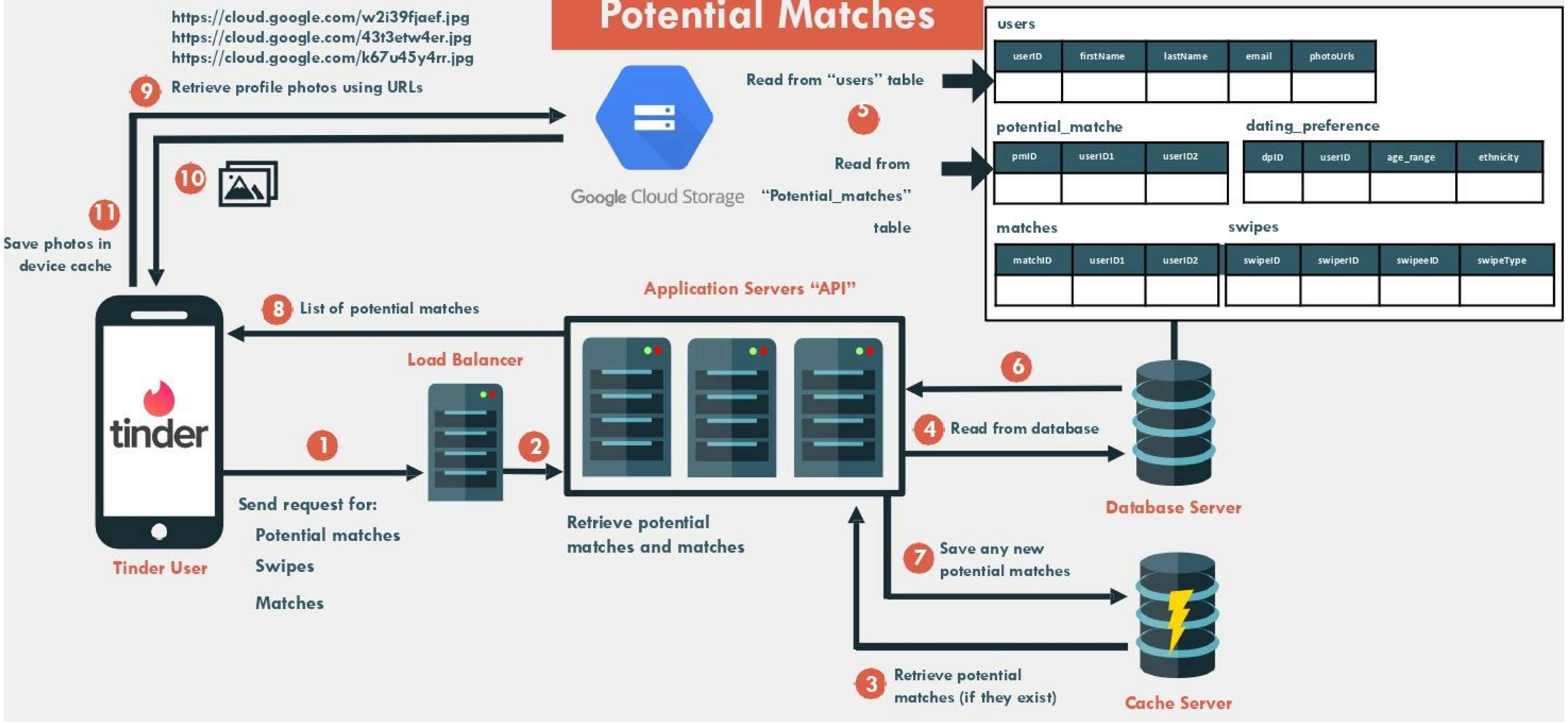
- Stores all of the matches
- Updates once both users have swiped right

matchID	user1ID	user2ID	timestamp
m1234	z1234	23r8w	2023-07-30 14:00:15

Registration


Write to "users" table
Write to "dating_preferences" table
users
userID
firstName
lastName
email
photoUrls
potential_matche
pmID
userID1
userID2
dating_preference
dpID
userID
age_range
ethnicity
matches
matchID
userID1
userID2
swipes
swipeID
swiperID
swipeeID
swipeType
Google Cloud Storage
https://cloud.google.com/w2i39fjaef.jpg
https://cloud.google.com/43t3etw4er.jpg
https://cloud.google.com/k67u45y4rr.jpg
3
Upload profile photo(s)
4
Send URL(s) for uploaded photo(s)
Application Servers "API"
Load Balancer
tinder
1
Send request for registration
2
5
Write to database
Upload photos to GCS
Store data in the database
Shrenik Surana
ss@gmail.com
[
https://cloud.google.com/w2i39fjaef.jpg,
https://cloud.google.com/43t3etw4er.jpg,
https://cloud.google.com/k67u45y4rr.jpg
]
Tinder User
Database Server

Potential Matches


https://cloud.google.com/w2i39fjaef.jpg
https://cloud.google.com/43t3etw4er.jpg
https://cloud.google.com/k67u45y4rr.jpg
9 Retrieve profile photos using URLs
10
11 Save photos in device cache
Google Cloud Storage
Read from "users" table
5
Read from "Potential_matches" table
tinder
Tinder User
8 List of potential matches
1
Send request for:
Potential matches
Swipes
Matches
Load Balancer
2
Application Servers "API"
Retrieve potential matches and matches
6
4 Read from database
Database Server
7 Save any new potential matches
3 Retrieve potential matches (if they exist)
Cache Server

users

userID	firstName	lastName	email	photoUrls

potential_matche

pmID	userID1	userID2

dating_preference

dpID	userID	age_range	ethnicity

matches

matchID	userID1	userID2

swipes

swipeID	swiperID	swipeeID	swipeType

APIs

Registration

Create a user (tinder.com/api/user/register)

➔ Email, password, profile photos

Get Potential Matches

Get all potential matches (tinder.com/api/potential-matches)

➔ User ID of user (might be optional)

Get Matches

Get all matches (tinder.com/api/matches)

➔ User ID of user (might be optional)

Swipe

Swipe (tinder.com/api/swipe)

➔ User ID of swiper (might be optional)

➔ User ID of swipee

➔ Swipe left or right