**MeetNLove Requirement Document**

1. **Features**

* User Signup / Login
* User Profile
* Personal Information
* Match Profile
* Chat / Messaging
* Likes You
* Subscription
* Settings
* Filters /User Preference
* Push Notification

1. **Microservices**

* Authentication-Service
* ProfileCreation-Service
* Notification-Service
* Recommendation-Service
* Swipe-Service
* Match-Service
* Payment-Service
* Chat-Service
* Subscriber-Service
* Blob-Storage / Image-Service
* Cache-Service

**In Scope:**

The application should be able to support the following requirements.

* User should be able to create their MeetNLove profile by adding their personal info and uploading photos.
* User should be able to view recommendations of other users in geographically nearby regions.
* Users should be able to like (swipe right) or dislike (swipe left) and super like other recommended users.
* User should chat other users to any location.
* Users should get notifications when matched with other users.
* Users should be able to filter user’s profiles based on location, age and preference.

**User Journey Detail:**

1. User Signup/Login –

* User click on login with mobile button. Once button is clicked user will redirected to Login Screen where he can enter his mobile number

System will send OTP to his mobile number. Once user received OTP, he can enter OTP. Once OTP is validated user will redirected to Profile creation page where he can fill his details

* After profile creation user redirected to Personal Info page where he can fill his personal details, interest and upload photos.

**Microservices details;**

1. **Authentication Service:**

* Create JWT auth token and validate every API call.

1. **User profile service:**

* This service will be invoked when user create his profile.
* This service will store user information in database.
* Store the user’s photos in Blob/File server.
* Publish user location (latitude/lang) to RabbitMQ/Kafka to store location info to Mongo DB/ Radis for Geospatial search which Recommendation service will used to find profiles.

1. **Recommendation Service:**

* When user view his recommendation profiles this service is called.
* Service will Forword request with (lat, lang, radius) to Geospatial (Mongo/Radis)
* Service will receive all response and apply filtering based on user preference and return the list of recommended users.

1. **Swipe Service:**

* If user A left swipe *SwipePublisher* publish this swipe to *leftswipequeue*
* Left swipe we can store into S3/Blob storage for further analysis
* If user right swipe *SwipePublisher* publish this swipe to *rightswipequeue* MatchService will received this request and queries against *LikedCacheDb* return the info if user B also right swipe and send notification to both the users. And also store this match to *MatchDB.*

1. **Match Service:**

* Match service will receive the right swipes data and query against LikeCacheDB if match found return the info and send push notification to both the users.
* If match not found then insert user key into database.

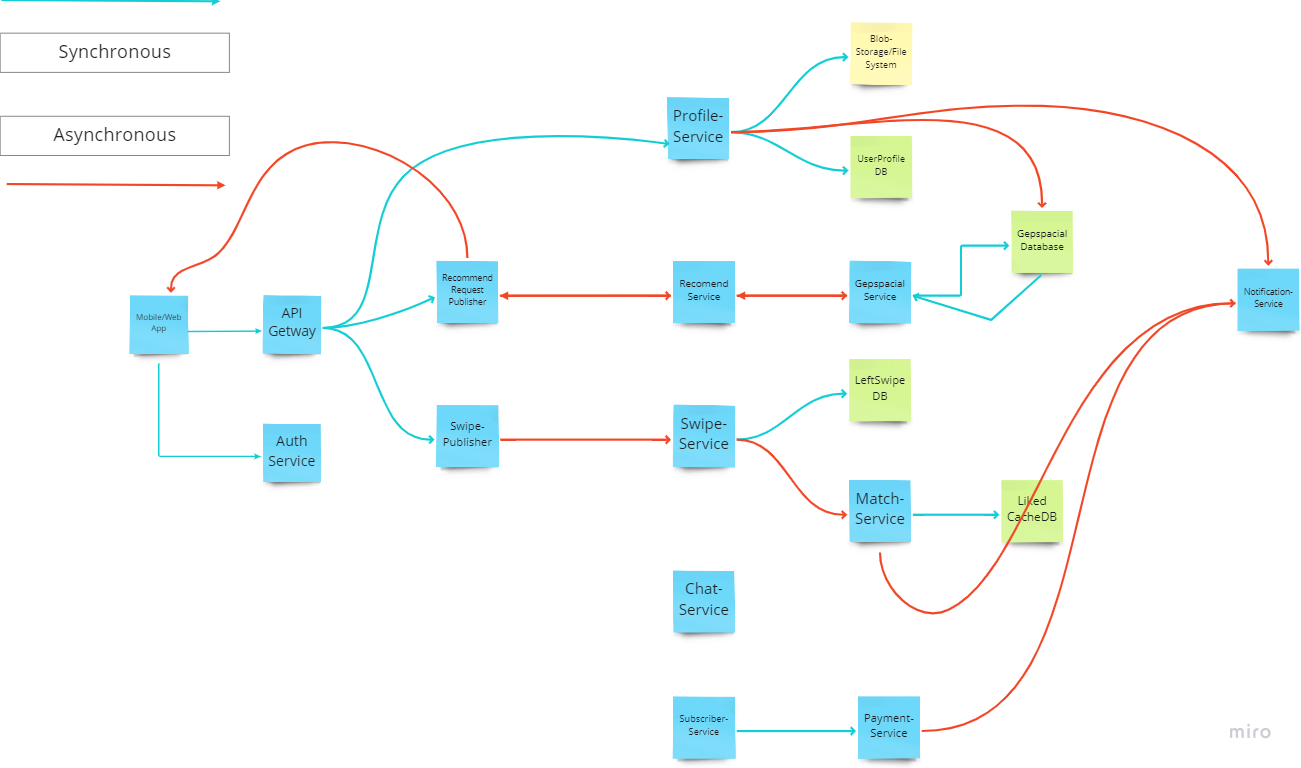
**Technology details:**

1. **Mobile Development:** Android Kotlin
2. **Authentication:** Oauth2.0
3. **Database:** MySQL, Mongo DB, Radis Cache
4. **Front-end:** Angular
5. **Programming Language:** .Net Core, C#, NodeJS,
6. **Cloud Services: Azure** Blob Storage, VM
7. **Chat:** SignalR, NodeJS
8. **Geolocation**: Google Map, Geospatial
9. **Monitoring and Logging:** Prometheus, Grafana
10. **Other Tools:** RabbitMQ, Dapper
11. **Push Notification:** Firebase, Twilio
12. **Payment Gateway**: Payu Integration
13. **Deployment:** Docker, Kubernetes
14. **Latency and Fault Tolerance:** Hystrix,
15. **API gateway:** Zuul
16. **Service Discovery & Load Balance:** Eureka server
17. **Microservice architecture:** [**https://steeltoe.io/microservices**](https://steeltoe.io/microservices)

**Useful Links:**

1. <https://learn.microsoft.com/en-us/shows/on-net/net-microservices-with-steeltoe>
2. <https://steeltoe.io/microservices>
3. <https://start.steeltoe.io/>
4. https://netflix.github.io/

**Application Architecture:**



**Application Brainstorming:**

