Languages: Java - great flexibility & portability - high productivity because of existing frameworks & previous knowledge

Spring - great portability/support in the cloud - easy and fast to bootstrap application with embedded application server - Component based systems (Beans etc.) - fast and uncomplicated support of REST-API features

ORM = SpringData - supports most DB-Technologies and reduces boiler plate code

Cloud platform/Server = Heroku - Ease of cloud deployment and continuous development - Free Account for development directly integrated with code sharing platforms like GitHub. Supports Continuous integration!

External Face Recognition API - SkyBioMetric(Free), LambdaLabs(Best support, but costly), OpenFace(OpenSource, more work)

Priorities - Security - Encryption and Authentication , sensible data - Portability/Integration & Usability - clear defined API - Reliability

Design Issues:

Used architectural patterns: Layered Approach - Access Layer (Client/UI/API), Database layer (ORM), Data .. Client/Server architecture - Separate development, Server provides API (cohesive service) - ¿ Service oriented architecture (External Service, ASM, USM) supports continuous development, integration and is very scalable. JSON Communication as standard MVC - Flexibility, Testability increased, separate components as Client/Server and separate logic and data etc.

1) Authentication - expires when? Usability vs. Security 2) Data storage - who stores data? PNs/Images Security vs. Performance/Cost 3) All in one component? - Extendability/Security vs. Developing Cost,Cost of component interaction(messaging...) 4)