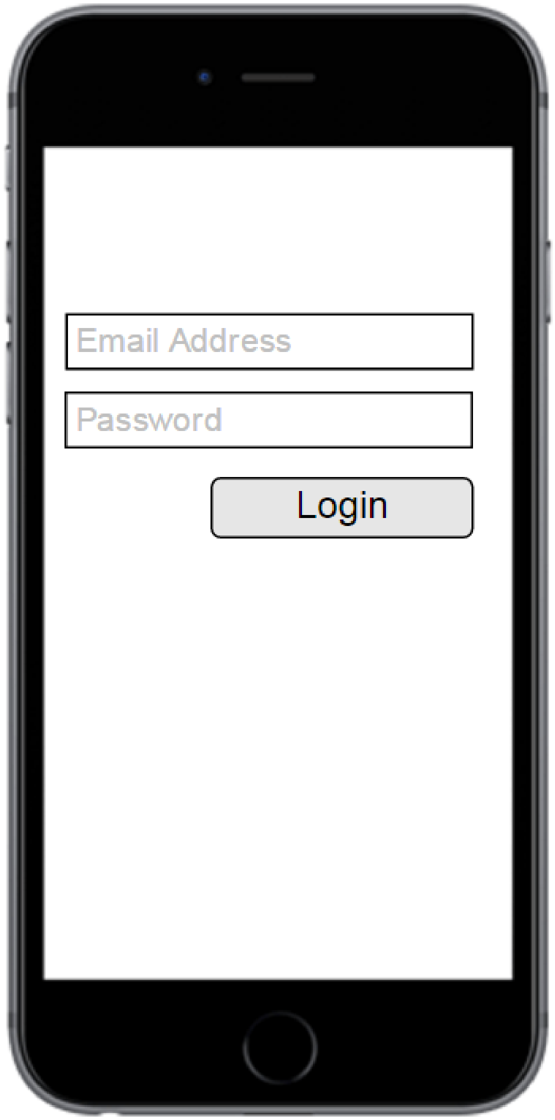
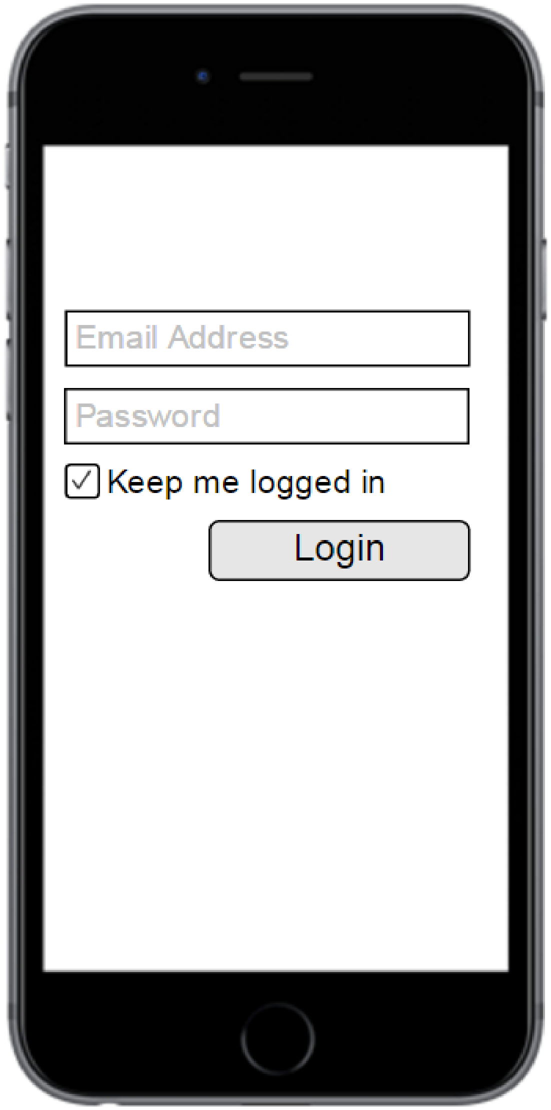
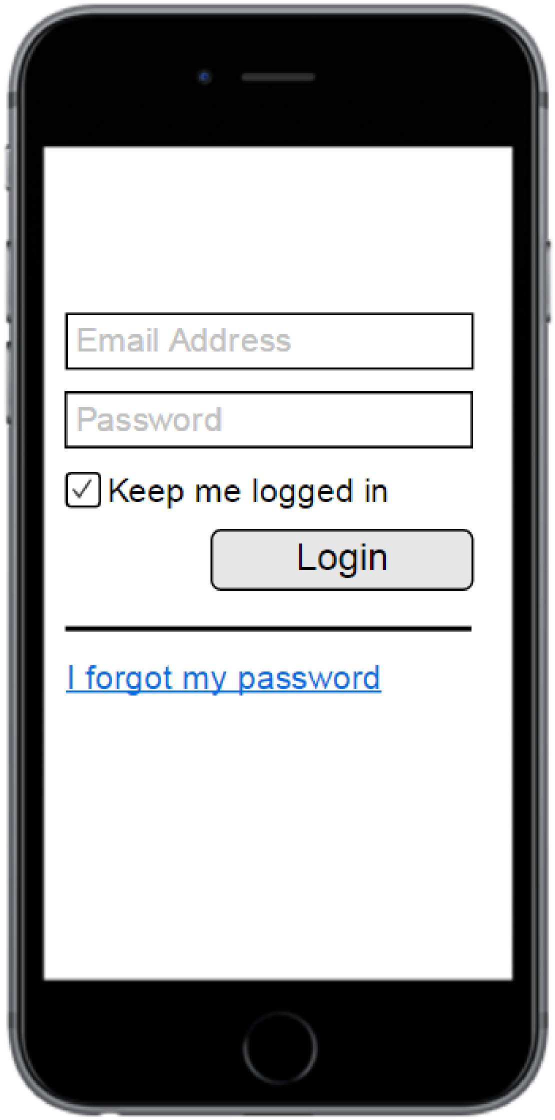
External software quality factors

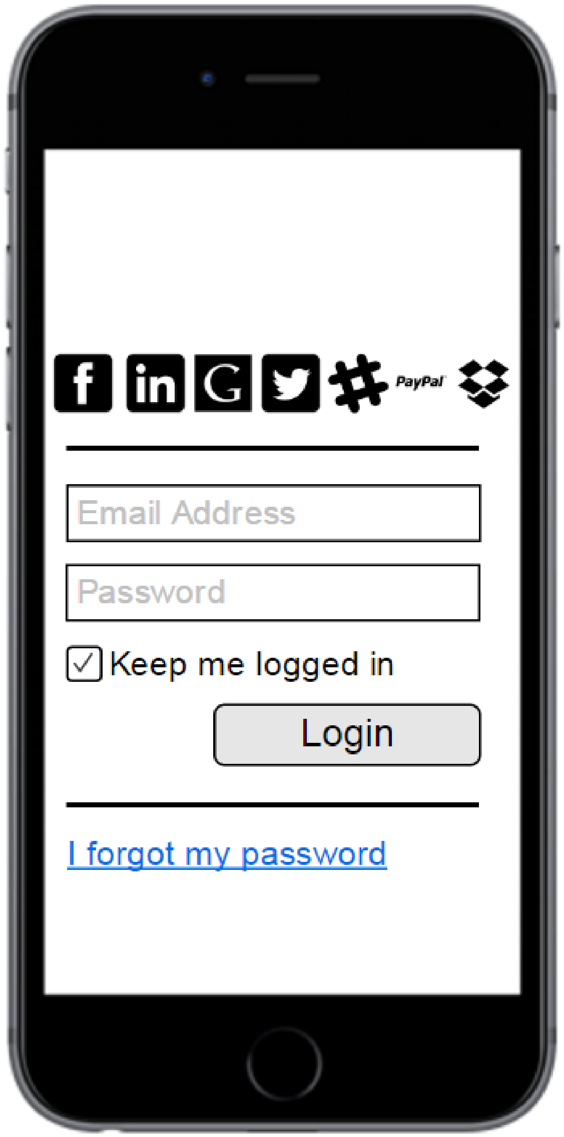
Evolution


Email Address
Password
Login


Email Address
Password
Keep me logged in
Login


Email Address
Password
Keep me logged in
Login
I forgot my password

Evolution



Email Address

Password

☑ Keep me logged in

Login

I forgot my password

Evolution

```cpp
void login(
        std::string const& email,
        std::string const& pass) {
}
```

```cpp
class provider {};
class service: public provider {};
class facebook: public provider {};
class linkedin: public provider {};
// etc.
template<bool const T>
void login(T rememberMe) {
}

template<class T, class... Args>
void login(T provider, Args... args) {
        // login with provider
        login(args...);
 }
```

Key factors

- Correctness (Корректность)
- Robustness (Устойчивость)
- Extendibility (Расширяемость)
- Reusability (Повторное использование)

- Compatibility (Совместимость)
- Efficiency (Эффективность)
- Portability (Переносимость)
- Easy of Use (Простота использования)
- Functionality (Функциональность)
- Timeliness (Своевременность)
- Verifiability (Верифицируемость)
- Integrity (Целостность)
- Repairability (Восстанавливаемость)
- Economy (Экономичность)

Design review check list

Correctness (Корректность)

- Do we cover all features from specification?
- Do we define clear borders between our system and external elements?

- Robustness (Устойчивость)

- Do we list all possible errors/emergency situations?
- Do we have clear plan how to handle, fix and restore after all this issues?

Design review check list

Extendibility (Расширяемость)

- Do we define which patterns and where we will implement?
- Do we plan a few design iterations?
- Do we have resources to pay technical debt?
- What is level of coupling for our system?

Reusability (Повторное использование)

- Do we use maximum number of open-source & proprietary ready-2-use solutions?
- Best code is code was not written.
- Do we have clear vision how and where we will reuse our components?

Design review check list

Compatibility (Совместимость)

- Do we use open/well-define/industry-proved standards?
- Do we follow SOLID principles?
- Patterns, again.

Efficiency (Эффективность)

- Do we define acceptable resonance time for all our communication channels?
- Do we define minimal hardware/software requirements?

Design review check list

Portability (Переносимость)

- Are we cross-platform?
- How may OS should we support, versions?
- Do we have DB abstraction layer?
- Do we design orthogonal APIs?

Easy of Use (Простота использования)

- Who are our users?
- How much time do we need to setup dev/qa/prod environments?
- Do we have auto generated source code documentation?
- What is size of user manual?

Design review check list

Functionality (Функциональность)

- Do we understand 80%/20% principle?
- Do we have prioritized back-log for current release? For one year from now?

Timeliness (Своевременность)

- Do we know our competitors?
- Do we have clear vision where domain goes?

Design review check list

Verifiability (Верифицируемость

- Do we plan to use TDD – Test Driven Development?
- Integration tests?
- Automated UI tests?
- HA tests?
- Logging & Audit

Integrity (Целостность)

- Security?

Design review check list

Repairability (Восстанавливаемость)

- Monitoring?

Economy (Экономичность)

- Estimates?