# 

# High-Level Design Document for Appstore project

|  |  |
| --- | --- |
| **Author** | Mohamad Badir, Waseem Sarsour, Wesam Hija |
| **Version** | V0.1 |
| **Created** | 5.5.2022 |
| **Last Updated** | 8.5.2022 |

# Introduction – this section specifies what you are actually implementing (usually subset of the functional specification)

## Assumptions and Design Constraints List

* Persistence of Blockchain records constrains the ability to delete and edit apps and their info.
* Assumption that various APIs can be integrated together to form a cohesive structure.

## Distributed & Parallel Aspects

What part do the distributed / parallel aspects play in the overall design and implementation of the project? Answer the questions that are relevant to your specific project in a paragraph(s) [don’t answer each question independently].

* What are the difficulties that are introduced when comparing to an equivalent monolithic (or sequential) implementation?
* How do you plan on dealing with the difficulties above?
* Does this project have an inherent distributed / parallel nature? Why?
* Why is this project interesting as a distributed system / parallel software? Are there any benefits?

The usage of Blockchain implies a distributed system from its nature. An additional parallel aspect is the review/download of a particular app, which may be done by multiple users at the same time.

## Dependencies

Describe dependencies of your project on other external features/libraries, along with backup plans in case of their unavailability.

Web3.js – to interact with the smart contract from the web app.

Truffle – for compiling/deployment. (backup – Hardhat)

Metamask – browser extension to access Ethereum wallet

And possibly others…

## Issues List

TBD

## To-do List + Expected time-tables

1. Finish learning smart contract, blockchain, react basics.
2. Setting up workspace environment for solidity.
3. Detailed planning of project structure (Classes, modules, etc)
4. Start implementing the UI for the application.
5. Write the smart contract that complies with the project’s requirements.
6. Write the web app logic code.
7. Start testing basic smart contract design.
8. Get familiar with qBitTorrent API.
9. Integrating various modules.

# Logical Architecture

Describe the architecture of your application. **Provide** a diagram like this one.

Diagram

Description automatically generated

## Module 1 [Webapp]

Website for browsing app catalog, can review apps and download the file transfer client.

## Module 2 [Metamask]

Facilitates communication between browser and blockchain.

## Module 3 [Blockchain]

Stores records of app info and metadata, including Torrent link and reviews.

## Module 4 [File Transfer Client]

Client downloaded from webapp, facilitates download/upload of apps via qBitTorrent API.

## Module 5 [qBitTorrent]

API for transfer of apps between clients.

## Module 6 [Truffle]

Deploys our smart contract to Blockchain.

# Design

## Modules / Main Classes

### Class Diagram

Class diagram of the main classes / modules.

Not yet planned.

### Class details

Details about the various classes / modules.

Not yet planned.

## Flows

Main flows for your application (you may use sequence diagrams).

* App author accesses webapp, downloads special program to upload his app along with necessary details and description. App finishes uploading and is added to website.
* User browses website. Can search or browse through apps. User chooses to install app, which downloads a file transfer program that sends the file to him. User can then rate and review the app.

## User interface(s)

Details about UI in high-level. For example, "a user will get login prompt and then get the main screen of the game ...", but not "login screen will contain OK/Cancel buttons and username field".

Upon accessing the web app, the user will get a login prompt where he can sign in. If he is an app author, he can access a dedicated page showing his apps and their reviews. If he is a normal user, he can view, download and review apps as detailed in the Functional Specification.

## Setup

Details about setup and installation (as much as you know by now).

The webapp can be accessed without an installation via a web browser. Installing or uploading apps downloads a special program to manage the file transfer.

# Use cases

Describe main use cases for your project. For the desired format, refer to slides on Intro to Software Engineering website. You are free, however, to choose any other acceptable format.

App authors can share their apps with others without having to host them on a dedicated server.

# References – to external papers/packages

Describe for each of them what you are using and what you are developing out of it.

If needed, indicate how it interfaces with the rest of the system.

TBD