

# **Addonis**

# The better add-ons registry

# Project Description

**Addonis** is an **Addons Registry** web application. Some of the possible actions it enables its users to do are:

* Publish their own addons
* Browse addons for their preferred IDE
* Download addons
* Rate existing addons

You may look for inspiration at the [JetBrains Registry](https://plugins.jetbrains.com/), the [Visual Studio Marketplace,](https://marketplace.visualstudio.com/) or the [Eclipse Marketplace](https://marketplace.eclipse.org/). The goal is to have a place that makes it easy for a user to find the right tool for their need. Build a clean, simple application that gets the job done!

# Functional Requirements

## Entities

* *Authentication is handled by Firebase, there is no need for auth entity*
* Each **user** must have a username, email, phone number and a photo.
  + Username must be unique and between 2 and 20 symbols.
  + Email must be valid email and unique in the system.
  + Phone number must be 10 digits and unique in the system.
* **Addon** must have a name, target IDE, creator, description, tags, binary content (the addon itself) and an origin (this is the location of the GitHub repo hosting the extension *(think of what additional entities you might relate to the Addon entity)*
  + Name must be unique and between 3 and 30 symbols.
  + The addon must have at least one tag (but can have more than one)
  + The addon must have exactly one target IDE.

## Public Part

The public part must be accessible without authentication i.e., for anonymous users.

Anonymous users must be able to register and login.

Anonymous users must see a landing page that has a list with three categories:

* **Featured** (addons selected by the admins)
* **Popular** (the addons with the most downloads)
* **New** (the newest addons)

Anonymous users must be able to still see details for each addon:

* Name
* Description
* Creator
* Tags
* Number of downloads
* Rating
* Download link
* Origin link – this is the location of the GitHub repo hosting the extension
* Open issues count (retrieved from origin repository)
* Pull requests count (retrieved from origin repository)
* Last commit (date + title) (retrieved from origin repository)

Anonymous users must be able to filter addons by name or IDE and sort by the following fields:

* Name
* Creator
* Tags
* Number of downloads
* Upload date
* Last commit date

Anonymous users must be able to download a plugin, directly from the landing page

## Private part

Accessible only if the user is authenticated.

Users must be able to login/logout, update their profile, manage their addons (CRUD).

Users must be able to view and edit their profile information, except their username, which is selected on registration and cannot be changed afterwards. The required fields for registration are username, email, and phone number.

Each user must be able to **rate** addons.

Once addon is created it must be in “pending” state until the administrator approves it. The extension is visible in the **Public** **Part** only if it is approved.

## Administrative part

Accessible to users with administrative privileges.

Admin users must be able to approve new addons

Admin users must be able to see list of all users and search them by phone number, username or email and block or unblock them. User list should support pagination. A blocked user must be able to do everything as normal user, except to create new and update current addons.

Admin users must be able to edit/delete all addons. Addons list should support pagination.

## Optional features

**Email Verification** – In order for the registration to be completed, the user must verify their email by clicking on a link send to their email by the application. Before verifying their email, users cannot make transactions.

**Addon Creation Verification** – In order to create an addon, the user is prompted to enter a verification code, sent to their email. The code should be unique for the addon and could have an expiration date.

**Invite a Friend** – A user can enter email of people, not yet registered for the application, and invite them to register. The application sends to that email a registration link.

**Identity Verification** – In order for the user registration to be completed, the user must submit a photo of their id card and a selfie. Users with administrator rights should have a page where they can view all users waiting for verification, review the photos they submitted and approve or reject them. Before being approved, users cannot create addons.

*Note:* ***DO NOT*** *upload actual photos of id cards!*

**Joint Addons** – User can create joint addons. They function as regular addons; however, multiple users can manage them. The owner of the addon has an administration panel for the addon, where they can grant or revoke other user’s access to the addon. Joint addon should show its owner and all its maintainers. Ownership could be transferred to a maintainer.

**Recurring Metrics Reports** – Users can set up recurring metrics reports. User has the option to select an interval of time on which the report is generated automatically. Reports should include all user’s addons (names) and their current number of downloads and rating. Users have a page, where they can view all their reports and cancel them. Users should be able to choose which addons to be included in the report. Users could be notified if a metric passes certain threshold (i.e., over 1000 downloads or rating falling below 3.5).

**Additional User Functionality** – Add search by creator in the public part. Any user should be able to follow other users for easy access to the list of their addons. User could be notified when a user from the following list creates new addon (after the verification). The user has a following list administration page, where they can remove users from the list.

**Drafts** – A user can create a draft of an addon. A draft addon is visible only for the creator. All data in a draft is optional but only a valid and complete addon can be switched to pending phase. Admins must not see drafts in the approve list but could have a separate list for all drafts in the system. Admins must only be able to delete drafts not to update them.

**Easter eggs** – Creativity is always welcome and appreciated. Find a way to add something fun and/or interesting, an Easter egg or two to you project to add some variety.

## Firebase Realtime Database

All data should be stored in the document (NoSQL) database hosted by Google Firebase. You must think of a way to organize your documents to achieve the functionalities described above.

# Technical Requirements

## General

* Follow [KISS](https://en.wikipedia.org/wiki/KISS_principle), [SOLID](https://en.wikipedia.org/wiki/SOLID), [DRY](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) principles when coding
* Follow the principles of functional programming wherever applicable
* Use tiered project structure (separate the application in layers, if applicable)
* You should implement proper exception handling and propagation
* Try to think ahead. When developing something, think – “How hard would it be to change/modify this later?”

## Git

Commits in the GitLab repository should give a good overview of how the project was developed, which features were created first and the people who contributed. Contributions from all team members must be evident through the git commit history! The repository must contain the complete application source code and any scripts (database scripts, for example).

Provide a link to a GitLab repository with the following information in the README.md file:

* + Project description
  + Link to the hosted project (if hosted online)
  + Instructions how to setup and run the project locally

## Optional Requirements

Besides all requirements marked as should and could, here are some more *optional* requirements:

* Use a branching while working with Git.
* Host you application with Firebase Hosting or any other hosting service

# Teamwork Guidelines

Please see the Teamwork Guidelines document.

# Appendix

* + [Git commits - an effective style guide](https://dev.to/pavlosisaris/git-commits-an-effective-style-guide-2kkn)
  + [How to Write a Git Commit Message](https://chris.beams.io/posts/git-commit/)

# Legend

* Must – Implement these first.
* Should – if you have time left, try to implement these.
* Could – only if you are ready with everything else give these a go.