Application Design and Software Structure Report

Alternative App Marketplace for cloud based ticketing software
Project codename: GiveThatDevACookie

Introduction / Features

Alternative app marketplace is a website which enables developers to advertise their apps by providing minimum information and effort.

Alternative app marketplace will consist, but not limited, of the following features:

- **Landing page**. Advertising featured apps on the landing page should improve SEO characteristics of the marketplace, should help visitors to see the most useful apps.
- **Contact page**. Allow contacting marketplace administrator to share any sort of feedback. This page will also contain information about the marketplace itself.
- **Apps page**. Will contain the list of apps including relevant filters. This enable visitors browse available apps.
- **Favorite apps page**. Will contain list of apps marked by login user. This enables visitors to quickly access the apps they like.
- **Share an app page**. Will enable login users to share their apps. This is how developers spread their ideas to the world.
- **App page**. Will contain details about selected app. This allow visitor to learn more about specific app as well as read and submit a feedback
- **Login**. Users will need to login in order to share their apps or share a feedback.

Design and Implementation

Application architecture is AngularJS based on the front-end level with nodejs server in the back end. Nodejs will act as a) REST API provider using MongoDB+Mongoose as a data base solution; b) server for AngularJS files.

The REST API Specification

Table below contains the summary of expected REST API end-points. The scope of expected end-points may be changed during the project development phase.

Table 1 – REST API end-points

#	Method	URL	Payload	Comment
1	GET	https://{host}/apps	n/a	Get all apps. Also supports search parameters in the URL: *?featured=true*
2	GET	https://{host}/apps/{app_id}	n/a	Get specific app by ID

3	POST
	[login:user]
		https://{host}/apps	{
			title: *string*,
			description: *string*,
			image_url: *string*,
			download_url: *string*,
			category: *string*,
			tags: *string*,
			featured: *boolean*
			}
				Create a new app record

4	DELETE [login:admin]	https://{host}/apps	n/a	Delete all apps

5	PUT
	[login:user]
		https://{host}/apps/{app_id}	{
			title: *string*,
			description: *string*,
			image_url: *string*,
			download_url: *string*,
			category: *string*,
			tags: *string*,
			featured: *boolean*
			}
				Update specific app

6	DELETE [login:user]	https://{host}/apps/{app_id}	n/a	Delete specific app
7	GET	https://{host}/apps/{app_id} /comments	n/a	Get all comments for specific app

8	POST
	[login:user]
		https://{host}/apps/{app_id}
		/comments
			{
			rating: *integer*,
			feedback: *string*
			}
				Create a new comment for specific app

9	DELETE [login:admin]	https://{host}/apps/{app_id} /comments	n/a	Delete all comment for specific app

10	GET	https://{host}/apps/{app_id}
		/comments/{comment_id}
			n/a	Get specific comment for specific app

11	PUT
	[login:user]
		https://{host}/apps/{app_id}
		/comments/{comment_id}
			{
			rating: *integer*,
			feedback: *string*
			}
				Update specific comment for specific app

12	DELETE
	[login:admin]
		https://{host}/apps/{app_id}
		/comments/{comment_id}
			n/a	Delete specific comment for specific app

13	GET [login:user]	https://{host}/favorites	n/a	Get all apps favorite by current user

14	POST
	[login:user]
		https://{host}/favorites	{
			_id: *string*
			}
				Add specific app into favorites

15	DELETE [login:user]	https://{host}/favorites	n/a	Delete all favorites
16	DELETE [login:user]	https://{host}/favorites/{app_id}	n/a	Delete specific app from the favorites
17	GET [login:admin]	https://{host}/users	n/a	Get all registered users

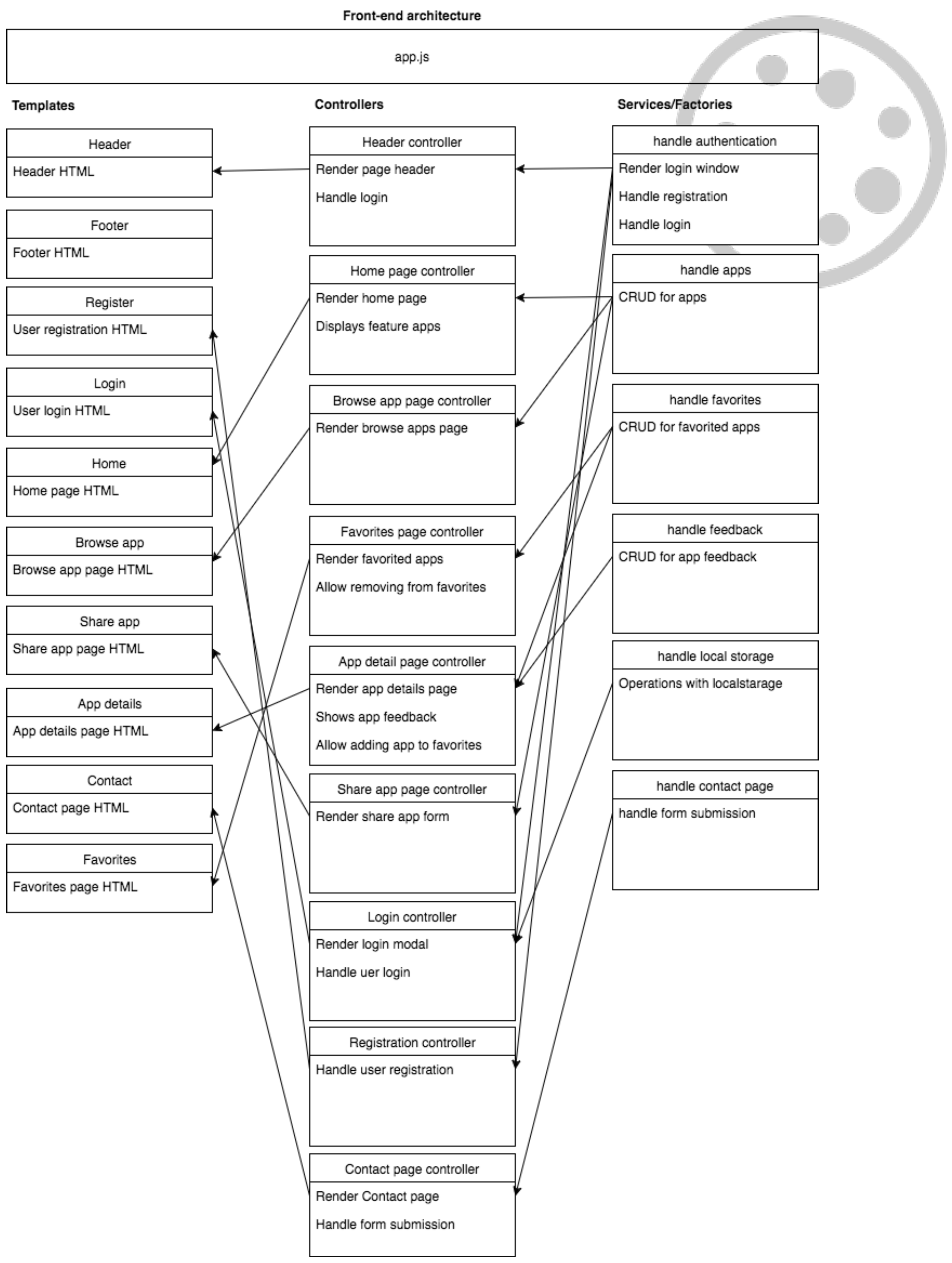
18	POST	https://{host}/users/register	{
			username: *string*,
			password: *string*,
			firstname: *string*,
			lastname: *string*
			}
				Register new user

19	POST	https://{host}/users/login	{
			username: *string*,
			password: *string*
			}
				Login user.
				Token will be issued in response. Token should be used in “x-access-

	token" header on further requests.

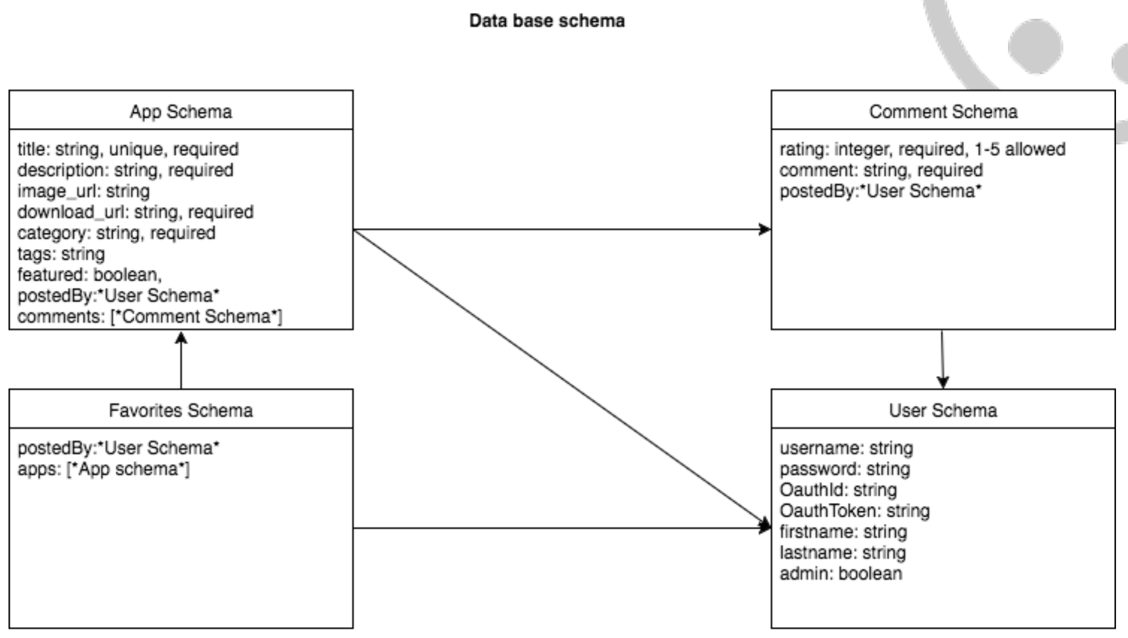
Front-end Architecture Design

Client-side will be handled by one of the first versions of AngularJS. Diagram below demonstrates the structure of the application broken down by templates, controllers and services/factories.


Front-end architecture
app.js
Templates
Controllers
Services/Factories
Header
Header HTML
Footer
Footer HTML
Register
User registration HTML
Login
User login HTML
Home
Home page HTML
Browse app
Browse app page HTML
Share app
Share app page HTML
App details
App details page HTML
Contact
Contact page HTML
Favorites
Favorites page HTML
Header controller
Render page header
Handle login
Home page controller
Render home page
Displays feature apps
Browse app page controller
Render browse apps page
Favorites page controller
Render favorited apps
Allow removing from favorites
App detail page controller
Render app details page
Shows app feedback
Allow adding app to favorites
Share app page controller
Render share app form
Login controller
Render login modal
Handle uer login
Registration controller
Handle user registration
Contact page controller
Render Contact page
Handle form submission
handle authentication
Render login window
Handle registration
Handle login
handle apps
CRUD for apps
handle favorites
CRUD for favorited apps
handle feedback
CRUD for app feedback
handle local storage
Operations with localstarage
handle contact page
handle form submission

Database Schemas, Design and Structure

Application data will be stored and handled using MongoDB with Mongoose. Diagram below represents expected database schemas and connections between them.



Communication

Client-server communication will be conducted usin HTTPS protocol. Table 2 below clarifies expected request structure.

Table 2 – Request structure

#	Attribute	Comment
1	Protocol	HTTPS is supported
2	URL	Example: https://{host}/apps/{app_id}
3	Method	GET, POST, PUT, DELETE is supported
4	Content-Type header	Content-Type:application/json
5	x-access-token header	x-access-token:{token_id} Access token received in server response after user login. For API end-point where authentication is not required x-access-token header can be ignored
6	Payload	Refer to REST API documentation for payload structure

Conclusions

Alternative app marketplace will be one page application driven by AngularJS v1 on the front-end level and supported by nodejs on back-end level. Nodejs will act as REST API provider and AngularJS files server. MongoDB and Mongoose will handle application data on the server side.

Suggested architecture will make app scalable and portable to mobile platforms. Using of AngularJS on front-end level allows fast conversion of web page into hybrid app using Ionic or similar frameworks. Using nodejs+MongoDB+Mongoose makes possible to quickly scale the application should web traffic increase. Wide range of REST API end-points creates number of further opportunities for further app improvements.

References

Diagrams produced using https://www.draw.io/

Zendesk® is a registered trademarks of https://www.zendesk.com

Cookie image used in this documented taken from https://thenounproject.com/term/cookies/122417