

Contents

1	Introduction	3
1.1	Background Study	3
1.1.1	Motivation	3
1.1.2	Relevance and Social Impact	4
1.1.3	Industrial Impact	4
1.2	Related Works	4
1.3	Problem Statement and Objectives	4
1.3.1	Problem Statement	4
1.3.2	Objectives	5
2	High-Level Design	6
2.1	Software Development Methodology	6
2.2	Architecture	7
2.3	RESTful API/End Points	7
2.3.1	GET Requests	7
2.3.2	POST Requests	7
2.3.3	PUT Requests	8
3	Detailed Design	10
3.1	Interface Design	10
3.1.1	The Users	10
3.1.2	Other softwares	10
3.2	Data Structures and Algorithms	10
3.2.1	createProprietarySoftware	10
3.2.2	getTopAlternatives	11
3.2.3	createFreeSoftware	11
3.2.4	getAllFreeSoftwares	11
3.2.5	checkLicense	11
3.2.6	getProprietarySoftwares	11
3.2.7	getAllProprietarySoftwares	11
3.2.8	getAlternatives	12
3.2.9	increaseUpvotes	12
3.2.10	decreaseUpvotes	12
3.3	UML Diagrams and Discussions	13
3.4	Data Source/Database used and formats	14
3.4.1	Details on Database	14
3.4.2	Data format/schemas	14

4	Implementation	16
4.1	Tools and Technologies	16
4.1.1	NodeJS	16
4.1.2	MongoDB	16
4.1.3	ReactJS	16
4.2	Coding Standards Followed	17
4.2.1	Proper Indentation / Structure	17
4.2.2	Naming Convention	17
4.2.3	Avoiding Callback Hell	17
4.2.4	Modular Approach	17
4.2.5	Proper Folder Structure/Naming	17
4.2.6	Model View Controller Model	17
4.2.7	Documentation	17
4.3	Execution Results and Discussions	18
5	Conclusion And Future Scope	25
5.1	Conclusions	25
5.2	Future Scope	25

Chapter 1

Introduction

“Free Software is software that respects your freedom and the social solidarity of your community. So it’s free as in freedom.”

Richard M Stallman

1.1 Background Study

The advent of technology brought great progress in various avenues of life. It has improved our living standards substantially and made it possible for us to achieve previously unimaginable gains. And needless to say, of past half a century or so, hardware coupled with software has brought infinite strength and weakness at our feet. We have landed on moon, sent machines to mars, digged ever so deeply into earth in search of gold and black gold alike and the list goes on. But what really has changed in the past three-four decades is the exponential growth of software.

1.1.1 Motivation

Software has brought in a massive revolutionary destruction and breakthrough which is giving us hopes to go beyond the imaginative horizons we now inhabit. We are now looking at self-driving cars, cure for cancers, early detection of deadly plagues and also, massive surveillance. And all of this is happening at an alarming rapid pace. The code we wrote yesterday is already obsolete today. And the code we are writing today is nothing short of miracle from yesterday. But that brings us to an important question : Where are we heading towards with it? And are we making sure that such rapid and reckless progress is not coming at the cost of freedom, privacy and safe home?

Our project here is aimed at providing a safe mind and a first step towards being free in this age of massive technological invasion by providing free software alternatives to ubiquitous proprietary softwares.

Free Software Movement, started back in 1984, is ever since marching relentlessly, braving all odds put up by those who are reaping unethical profits from software they write. Our project is aiming to add a bit of our own share towards the movement. Many a times, we are aware that the softwares we are using are proprietary by nature and thus we have no control over it whatsoever. We are forced to accept what comes packed in a pink ribbon without having the leeway to ask for changes or wish for betterment. We are being traded with convenience in return for our privacy because we don’t know what’s going on behind the scenes. So a safer and much better situation would be to resort to free softwares that are developed by communities that we can either directly be part of or can trust them because we can see for ourselves very transparently as to what the communities are doing to and with the software they spawn.

1.1.2 Relevance and Social Impact

The above concerns and issues so raised can be rounded about by a platform like AlterFoss. As software engineers, we are both consumers as well producers of software. And in the current world, the role of software and the power it wields is ever so immense and massive respectively. Our code, which is merely a result but also an important proxy of the design that goes behind and many technical and non-technical decisions that were employed during the construction, can now be seen seeking refuge in every domain of our lives. Many a times, it's part of time and/or life critical systems such as medical equipments, self driving cars and of course, our gadgets. It's omnipresence makes it very potent which then leads to an obvious fork in our lives : how can this power be harvested for betterment of humanity and not let it go against it?

When our code is hidden from the world, we as developers have all the freedom to employ it as we deem fit. Thus a serious leak of power in the wrong streams. This is how it's going on so far. Most of the titans of the tech industry follow the proprietary model of software development which essentially means we aren't allowed to see what's under the hood. This is rather worrisome.

We the developers have a moral responsibility to not subject our users to such injustice. We shouldn't misplace their trust. That's where our project finds most of its relevance. Also, it's important that early on in our careers (which really start on day 1 of our engineering course), we start contributing to free and open source projects so that we get practical as well as emotional relevance in the tech world.

Our project offers a convenient platform for users to search and find appropriate free and/or open source alternatives to the proprietary softwares they are using. This is helpful to various kind of users. The non-developer community can simply seek alternatives for use whereas developers can run searches on our platform to find projects which they think fit to their skill-set as well as interests.

1.1.3 Industrial Impact

Upto 53% of companiils across industries have switched/are switching towards free and open source software systems and/or models. Within the software industry, as much as 85% of companies have a project under free and/or open source softwares or are using products of such a model. That's very encouraging for developers like us who stage a belief in FOSS. For industries too, our platform can be used to find and work on free softwares that they see fit for their uniques circumstances. We may not be reinventing the wheel but we are certainly pulling in a work that can make a difference for all it's users. Industries are driven by engineers, and engineers with their work and talent invested in right places are indisputable.

1.2 Related Works

- Free Software Organization : An organization by Richard Stallman that has indexed all the popular free and open source softwares
- AlternativeTo : An online web application that gives alternatives of any software indexed by it. It provides any kind of alternative without any restrictions.

1.3 Problem Statement and Objectives

1.3.1 Problem Statement

AlterFoss is a web-based application that provides information on the alternatives to proprietary softwares. Users who are now willing to take their privacy in their own hands can make the best use of our platform but all users can do so nevertheless. The platform provides appropriate and quick response to a user query. As cited already, alternatives are presented to the users on demand.

1.3.2 Objectives

- To provide a search for proprietary software and get its corresponding FOSS alternatives.
- To allow to like/dislike the alternatives.
- To provide an interface to raise a request for alternatives by providing proprietary software details.
- To provide an interface for users to suggest alternatives to already present proprietary softwares.
- Develop user authentication for the above three tasks.
- Develop straightforward and easy to use UI
- Develop robust back-end support
- Develop a platform that community can use and grow on its own.

Chapter 2

High-Level Design

”Software development is not a rational process. It’s a process made by people with feelings, with bodies and with thinking. And by putting all those together I can be a more effective software engineer.”

Kent Beck

2.1 Software Development Methodology

XP Workflow

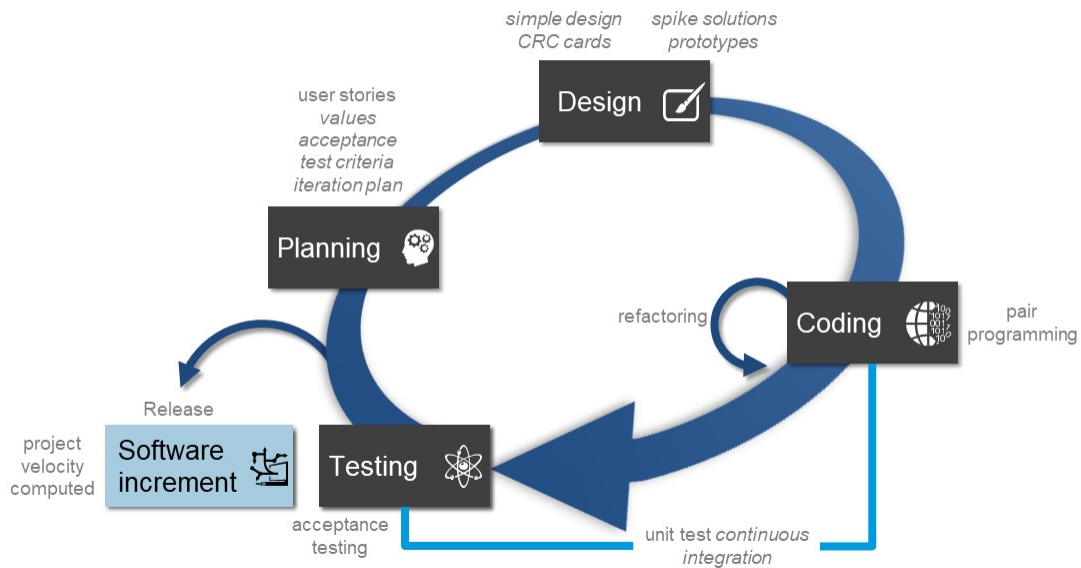


Figure 2.1: XP Workflow

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it

advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers.

The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed continuously, i.e. the practice of pair programming.

2.2 Architecture

2.3 RESTful API/End Points

Each alternative software will be initialized with 0 upvotes and downvotes. These fields will be via `req.body.<field-name>`. If the `suggestedBy` field is not specified, then the default value is taken to be "anonymous". If the input syntax does not match this standard, the returned value is a string specifying the error.

2.3.1 GET Requests

- 'GET' /
Home Page
- 'GET' /api/login/me
Used to query the database for user information. A token needs to be sent in the header of the request which shall then be verified. On successful varification, usermae, firstName and lastName are returned in response. Otherwise a 400 error is raised.
- 'GET' /api/proprietary
Returns an array of all proprietary softwares as an array of objects.
- 'GET' /api/alternatives
Return an array of the top 10 alternatives (Free Softwares) as an array of objects containing the name, license and upVotes for the softwares in decreasing order.
- 'GET' /api/alternatives/<id>
Returns the alternatives for the proprietary software specified by id. id has to be passed as a in the url.

2.3.2 POST Requests

- 'POST' /api/alternatives
To add a new alternative to the free softwares. The body of the request contains the following fields:
 - **name**: The name of the software
 - **shortDescription**: A one line description of it
 - **handle**: A single keyword which will be used to assign it as an alternative
 - **license**: The software's license
 - **suggestedBy**: The username of the user who proposed this alternative

- 'POST' /api/proprietary/

This will add a new proprietary software. Similar to adding alternative softwares, here also the request body must contain the following fields:

- **name**: name of the proprietary software
- **shortDescription**: One line description of the software
- **tags**: A string array consisting of strings that will be matched against the handle of the alternative softwares to verify it is an alternative
- **requestedBy**: Username of the user who put up this proprietary software request

If the **requestedBy** field is not specified, then the default value is taken to be "*anonymous*". If the input syntax does not match this standard, the returned value is a string specifying the error.

- 'POST' /api/proprietary/

Used to search for proprietary softwares. The pattern will be used as a regular expression to search for proprietary softwares. Pass the search string the user enters to this endpoint. Returns an array of objects corresponding to the proprietary softwares matching the given search string. The string will be passed as a property 'search' inside the **req.body**.

- 'POST' /api/alternatives/license/

Used to search alternative softwares to check their license. Returns an array of objects containing the name of the alternatives matching the search property of the **req.body** object. The matching is done as **regExp** matching ignoring case.

- 'POST' /api/proprietary/search/

Used to search proprietary softwares. Returns an array of objects containing the name and short-Description of the proprietary softwares matching the search property of the **req.body** object. The matching is done as **regExp** matching ignoring case.

- 'POST' /api/signup

Used to sign-up a user onto the application. It takes user credentials in the request's body and then validates for correctness. If validation passes, the user is stored onto the database and automatically logged in. Additionally, a json web token is sent for future use. If the validation fails, the user is prompted with appropriate error message.

- 'POST' /api/login

Used to log-in a user onto the application. There are two ways of doing it :

1. **Through log-in credentials** : This method involves user sending in his uername/email and password. On successful validation of the credentials, user is logged onto the system as well as a json web token is sent for future use(session handling).
2. **Through json web token** : If a user queries with a valid json web token, he is logged onto the system after varifying the token's authenticity. Otherwise an **invalid-token** error is raised.

2.3.3 PUT Requests

- 'PUT' /api/alternatives/upvote/<id> This will increase number of upvotes for the given alternative software specified by id by 1. The id has to be passed as a parameter in the url.
- 'PUT' /api/alternatives/unupvote/<id> Will reduce the number of upvotes for the given alternative software specified by id by 1. The id has to be passed as a parameter in the url. Automatically checks if the upvotes is already at 0, doesn't do anything.

- 'PUT' /api/alternatives/downvote/<id> This will increase number of downvotes for the given alternative software specified by id by 1. The id has to be passed as a parameter in the url.
- 'PUT' /api/alternatives/undownvote/<id> Will reduce the number of downvotes for the given alternative software specified by id by 1. The id has to be passed as a parameter in the url. Automatically checks if the upvotes is already at 0, doesn't do anything.

Chapter 3

Detailed Design

3.1 Interface Design

This section deals with how this software interacts with people, hardware and other software.

3.1.1 The Users

The webApp interacts with users using UI components designed on React. The users can enter text in the text boxes for various purposes such as searching, sending information regarding free softwares and proprietary softwares and obtaining license information.

3.1.2 Other softwares

Various modules have been implemented and imported throughout the project to enable different software components to interact with each other. Modules such as 'express' and 'mongoose' have been used to connect api endpoints with implemented functions and connect to the database respectively. The back-end implemented on nodejs is integrated with the front end react components using RESTful API.

3.2 Data Structures and Algorithms

This section deals with various functions implemented along with their purposes and the data structures used.

3.2.1 createProprietarySoftware

This function is used to create a new proprietary software by taking information from the user-end. This function will only be accessible when the user searches for a proprietary software and its not available already. Then if the user wishes to request for the proprietary software, he can submit the request consisting of the name of the proprietary software along with other details such as a short description regarding the software along with a tag that will be used while searching for alternatives for this software.

A json object will be used to send data regarding this software using key-value pairs to the back-end to process and add this data to the database.

A input validation happens at the database level and the user is prompted with a input error if the format of the input does not match the required specification.

3.2.2 `getTopAlternatives`

This function is used when the user wants to know the trending alternatives with the maximum upvotes. This function returns an array of json objects containing the properties of the top 10 alternatives sorted in descending order of their upvotes. The user does not need to specify any input for this function as it directly fetches data from the database.

In case the number of documents in the alternatives section of the database is less than 10, all the alternatives are returned in descending order. However, if there's no data in the alternative softwares section this function returns 0, which can be further used to display relevant error messages to the user.

3.2.3 `createFreeSoftware`

Similar to the `addProprietarySoftware`, this function is used to add alternative software for a specific proprietary software. The user needs to specify the name, short description, license fields of the alternative software. The handle field of the alternative json object is set using the tags field of the proprietary software for which the alternative is being suggested.

A json object will be used to send data regarding this software using key-value pairs to the back-end to process and add this data to the database.

An input validation happens at the database level and the user is prompted with an input error if the format of the input does not match the required specification.

3.2.4 `getAllFreeSoftwares`

This function fetches all the free softwares from the database along with their short descriptions, upvotes and license. It will be used in case the user wishes to see all the alternatives which are available on the website and this returned value can be used for performing various other manipulations on the retrieved data which can be used to meet the needs of some other functions which can be implemented in future. An array of json objects will be returned by this function containing properties like name and a short description, license and upvotes of the software.

No input validation is required as the function does not take any input. However, if there are no free softwares present in the database, the function returns a suitable error message.

3.2.5 `checkLicense`

This function takes a pattern as parameter which will be used to match it to the names of the alternatives in the database and returns an array of objects containing the license information of all the matching results. The matching is performed by treating the pattern as a `RegExp` ignoring the case.

If the pattern doesn't match any alternative in the database, a suitable error message is returned.

3.2.6 `getProprietarySoftwares`

This function takes a pattern as parameter which will be used to match it to the names of the proprietary softwares in the database and returns an array of objects containing the name, short description and tags information of all the matching results. The matching is performed by treating the pattern as a `RegExp` ignoring the case.

If the pattern doesn't match any proprietary software in the database, a suitable error message is returned.

3.2.7 `getAllProprietarySoftwares`

This function fetches all the proprietary softwares from the database along with their short descriptions. It will be used in case the user wishes to see all the proprietary softwares which are available on the website and this returned value can be used for performing various other manipulations on the retrieved data which can be used to meet the needs of some other functions which can be implemented in future. An array of json

objects will be returned by this function containing properties like name and a short description, license and upvotes of the software.

No input validation is required as the function does not take any input. However, if there are no free softwares present in the database, the function returns a suitable error message.

3.2.8 getAlternatives

This function takes in the id of a proprietary software in the database and returns an array of json objects of alternatives to this particular proprietary software (corresponding to the id). The handle of the alternative is searched for in the tags field of the proprietary softwares. If found, the alternative is concluded to be an alternative of this proprietary software and is added to the array of results which will be returned by this function.

If no proprietary software is found for the specified id, the function will return a suitable error message.

3.2.9 increaseUpvotes

This function simply increases the upvotes property of the alternative software specified by its id which will be specified as a parameter to this function.

In case no alternative is found corresponding to the passed id, a suitable error message is returned by this function.

3.2.10 decreaseUpvotes

This function simply decreases the upvotes property of the alternative software specified by its id which will be specified as a parameter to this function.

In case no alternative is found corresponding to the passed id, a suitable error message is returned by this function.

3.3 UML Diagrams and Discussions

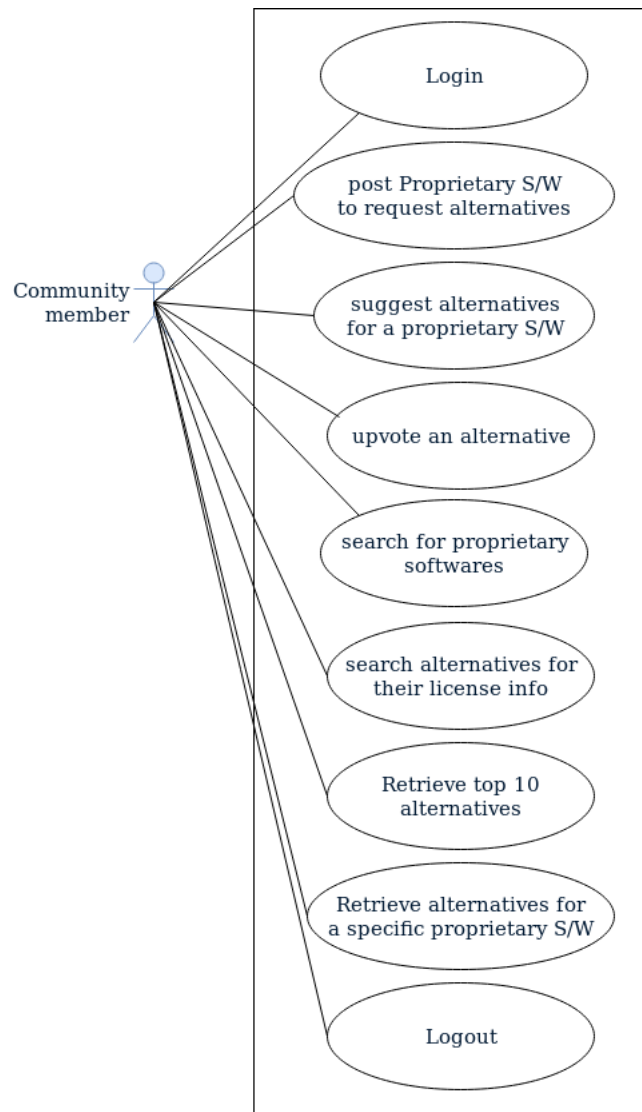


Figure 3.1: User-case diagram of AlterFoss

As shown above in the use-case diagram, this entire platform will be community driven. Once logged in, the regular community members will have the facilities to post new proprietary softwares to request for its alternatives, suggest alternatives for already posted requests (proprietary softwares) as well as get already posted alternatives. The users can search for proprietary softwares and upvote specific alternatives. The users can also retrieve the top 10 voted alternatives and check the license information of all posted alternatives. Also, searching and license checking do not require logging in.

3.4 Data Source/Database used and formats

This section deals with the database used and the format in which the data is stored in the database.

3.4.1 Details on Database

The database used for this project is **MongoDB** which is a free and open-source cross-platform document-oriented database program. MongoDB uses JSON-like documents with schemas and is classified as a NOSQL database. It is published under a combination of the GNU Affero General Public License and the Apache License.

3.4.2 Data format/schemas

There are majorly 3 entities used in the project:

- proprietarysoftware
- freesoftwares
- usercredentials

Each of these entities are being stored as mongodb documents. The detailed structure/format of all these entities is as follows:

Proprietarysoftware

This schema contains the following fields :

name : A string used to store the name of the proprietary software. This field is required and can't be set to empty.

shortDescription : A string used to store a single line description for the proprietary software. This field is also required.

tags : An array of strings used to characterize the proprietary software. These strings can be used to find suitable alternatives for this proprietary software. This field also has a validator function set to make sure that the user specifies at least one tag for every proprietary software.

requestedBy : A string used to store the name of the user who posted this proprietary software to request for its alternatives. If no value is specified for this field during the creation of the proprietary software, "anonymous" is taken as the default value.

Freesoftware

This schema contains the following fields :

name : A string used to store the name of the alternative. This field is required and can't be set to empty.

shortDescription : A string used to store a single line description for the alternative. This field is also required.

upVotes : An integer to store the number of upvotes of the alternative. Can be used by the community to evaluate the quality of the alternative. Initialized to zero initially.

- downVotes :** An integer to store the number of downvotes of the alternative. Can be used by the community to evaluate the quality of the alternative. This property is currently not being used but will be considered for future functionality. Initialized to zero initially.
- handle :** A string which will characterize the alternative. This property will be used to find which proprietary software this alternative is fit for. This field is probably the most important one as this is the one which will pair it with its proprietary counterpart.
- license :** A string which stores the name of the license under which this software was published. This field is required thus it cannot be empty.
- SuggestedBy :** A string which stores the name of the user who suggested this alternative. If no value is specified for this field during the creation of the proprietary software, “*anonymous*” is taken as the default value.

Usercredentials

This schema contains the following fields :

- firstName :** A string used to store the first name of the user. This field is required and can be a combination of alphabets with a maximum length of 50. This specification is validated by using a regExp for this field.
- lastName :** A string to store the last name of the user. This field is required and can be a combination of alphabets with a maximum length of 50. This specification is validated by using a regExp for this field.
- gender :** A string to store the gender of the user, which can be one of the three enumerated values: male, female, or something-else.
- userName:** A string to store the username of the user. This can be a combination of alphanumeric characters and - with a minimum length of 3 and a maximum length of 30 as specified by the regExp validating this field. This field must be unique for every user.
- password :** A string to store the hash of the user’s password. This too is a required field.
- email :** A string to store the e-mail of the user. This field is required and must be unique for every user. This field is also validated by a regular expression and can have a maximum length of 255.
- hasUpvoted :** An array to store the ids of the alternatives for which the user has upvoted. This can be used to verify that the user cannot upvote the software for which he has already upvoted.

Chapter 4

Implementation

This section deals with the realization of technical specification , design, software component and standard algorithm. It includes methods which are implementations of those methods specified by the interface.

4.1 Tools and Technologies

4.1.1 NodeJS

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

NodeJS has been used to write the complete back-end of our mini-project for creating RESTful APIs that are used to integrate back-end with the front.

4.1.2 MongoDB

The database used for this project is MongoDB which is a free and open-source cross-platform document-oriented database program. MongoDB uses JSON-like documents with schemas and is classified as a NOSQL database. It is published under a combination of the GNU Affero General Public License and the Apache License.

This is a NOSQL database so our need is easily fulfilled using this database as it stores data in document format using key-value pairs and js also works on json objects, thus simplifying the task of retrieving and manipulating the data.

4.1.3 ReactJS

ReactJS is a javascript library which is used for creating user interfaces. React makes it easy to maintain the front-end. As our project is a modular based project where all the features have been implemented as different modules, react, which uses a component based approach to developing user interfaces, can go hand-in-hand with the back-end. Due to this very nature of react, we can easily scale it in future too. Moreover, react is very lightweight thus takes less loading time. Hence, it's really convenient for designing our project's user Interface.

4.2 Coding Standards Followed

4.2.1 Proper Indentation / Structure

The entire code is properly indented with uniform spaces and also leaving spaces between operands and operators. All the modules to be imported are imported at the top of the files, thus making the code clearer and easy to understand.

4.2.2 Naming Convention

The variables are given meaningful names and naming is performed using camel case variable naming convention. This makes the code neater and the code becomes more meaningful and easy to understand.

4.2.3 Avoiding Callback Hell

All the functions written inside the controller are designed using the async-await approach using promises, thus avoiding callbacks almost completely. This makes the code much more linear avoiding the deeply nested structure which results due to using callbacks, called the callback hell. We have successfully avoided that problem by writing all the functions using async-await approach.

4.2.4 Modular Approach

Each feature has been implemented as a completely independent function which is exported by the controller module. The functioning of each and every one of these functions does not depend or effect the functioning of other functions. This makes the addition and removal of new features easy. Also, debugging has been made easy due to this approach.

4.2.5 Proper Folder Structure/Naming

A standard folder structure has been followed in the project, keeping all the related files in different folders. The files containing the core functions are kept inside controller folder, all the api end-point containing files are kept inside routes folder, all the database schema models are kept inside models folder.

4.2.6 Model View Controller Model

Model–View–Controller (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

4.2.7 Documentation

All the API end-points have been properly documented to make the integration easier and clear. The documentation contains the type of request to be sent to a specified route along with all the required input and the expected output along with all the constraints and error possibilities.

4.3 Execution Results and Discussions

This section shows the execution results after integration and deployment. We shall be looking at some of the screenshots of the user interface which will outline the functioning of our app.

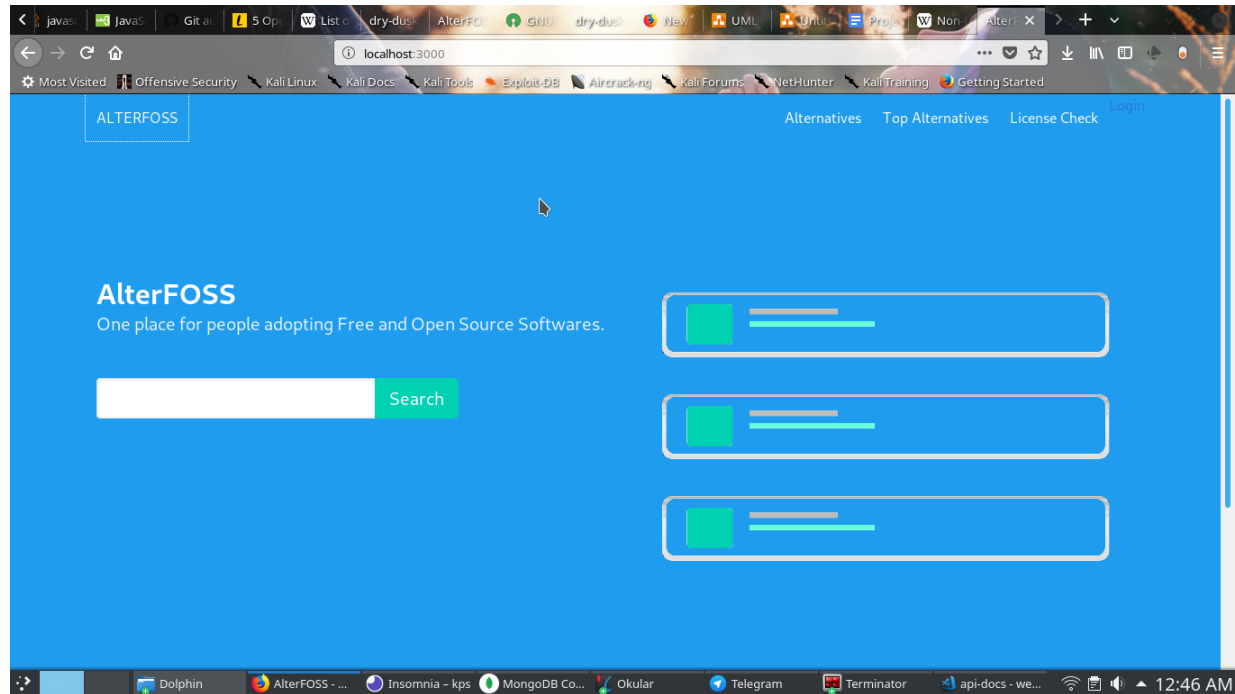


Figure 4.1: Home Page

The figure shown above is the home page of our web Application. The various links can be seen above such as alternatives to get alternatives for proprietary softwares, top alternatives which is used to list out the top 10 alternatives based on upvotes. License check is used to check the license information of alternatives. The search bar shown here is used to search proprietary softwares based on the text the user passes via this text box.

The below figure (fig. 4.2) shows proprietary search results for search query “windows”. This search is performed by treating the input text as regular expression. The information displayed is the name of the proprietary software along with the description of the software in the order in which it was found.

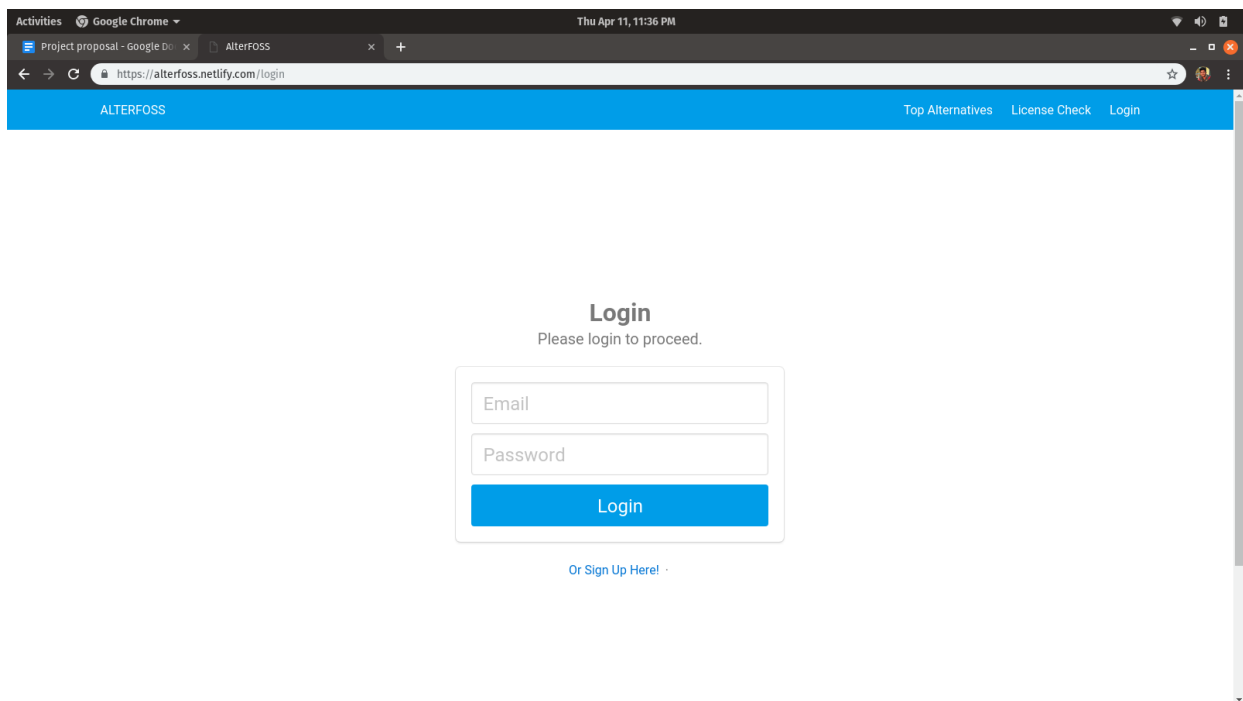


Figure 4.2: Home Page

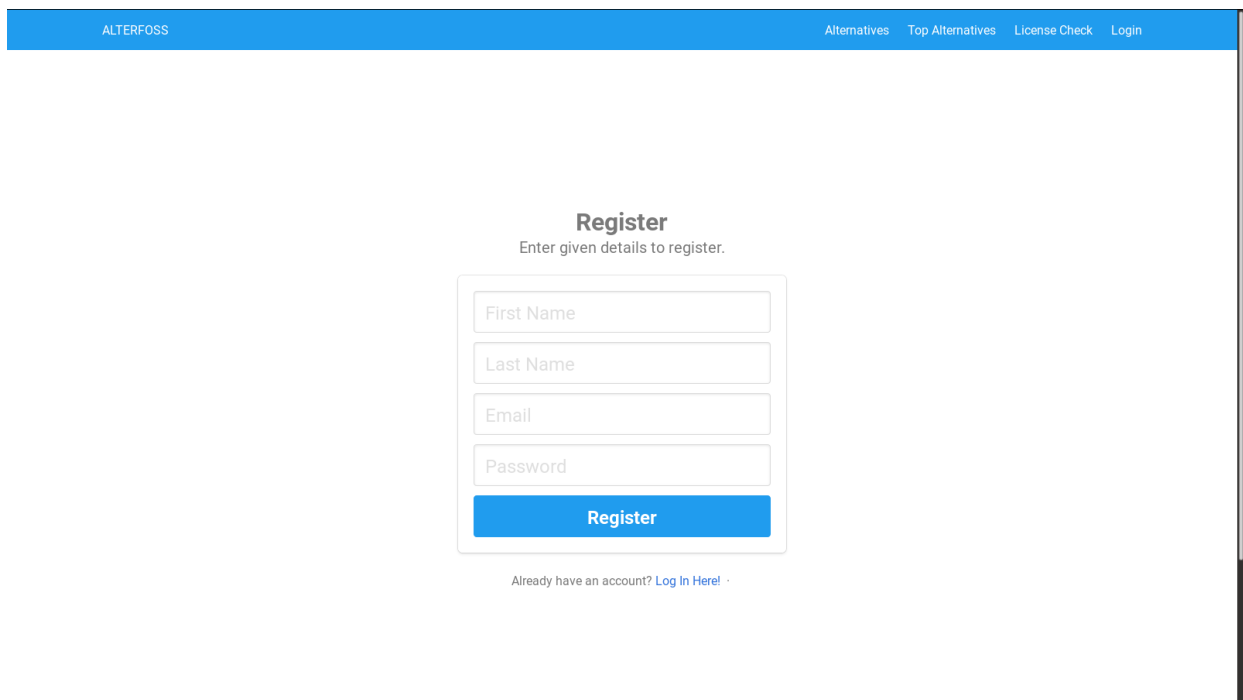


Figure 4.3: Home Page

Figures 4.2 and 4.3 show the authentication interfaces of the application.

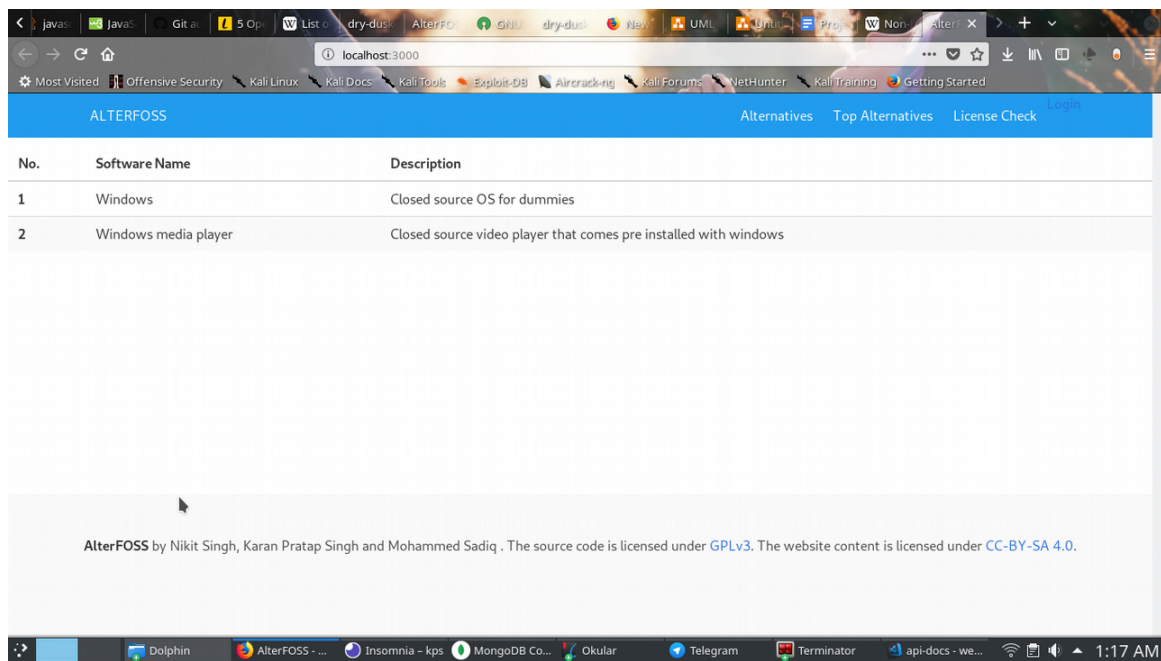


Figure 4.4: Proprietary Search Results

In fig. 4.4, we had searched for 'windows media player' and those were the replies we got.

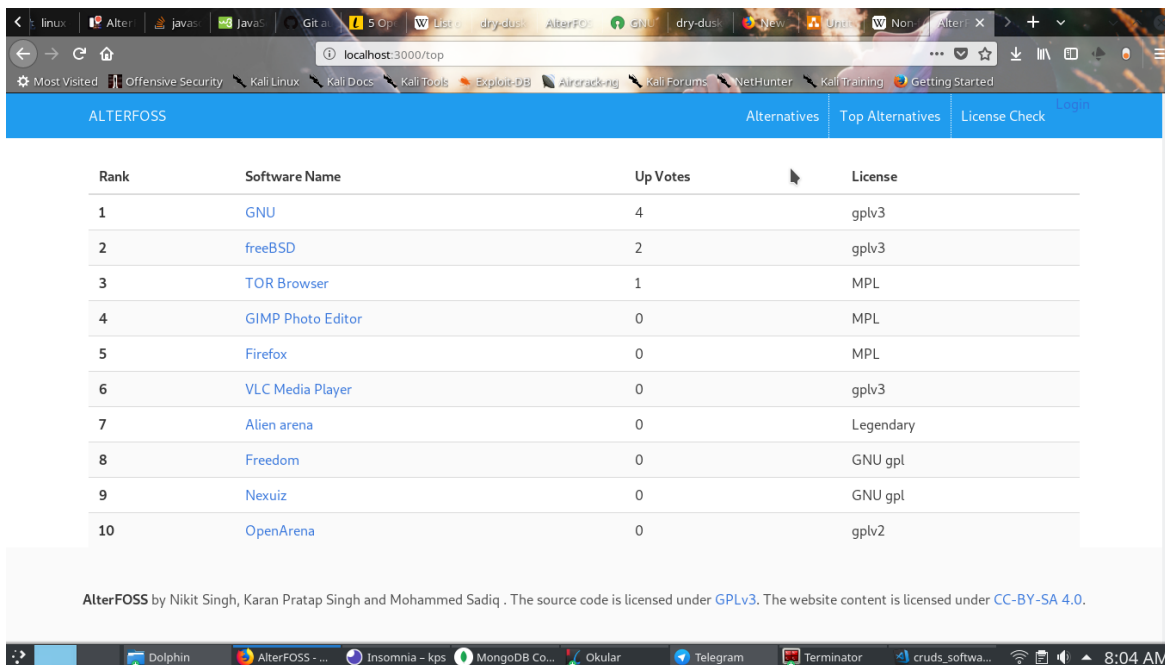


Figure 4.5: Top 10 alternatives

As shown in fig. 4.5, the top 10 alternatives list can be viewed by the user by clicking on top alternatives option on the top right of the web page. It will display the list of top 10 alternatives along with their upvotes and license. This list is sorted, as discussed previously, based on the number of upvotes of the software. The api fetches the top 10 alternatives from the database, sorts them and sends them to the front end for display.

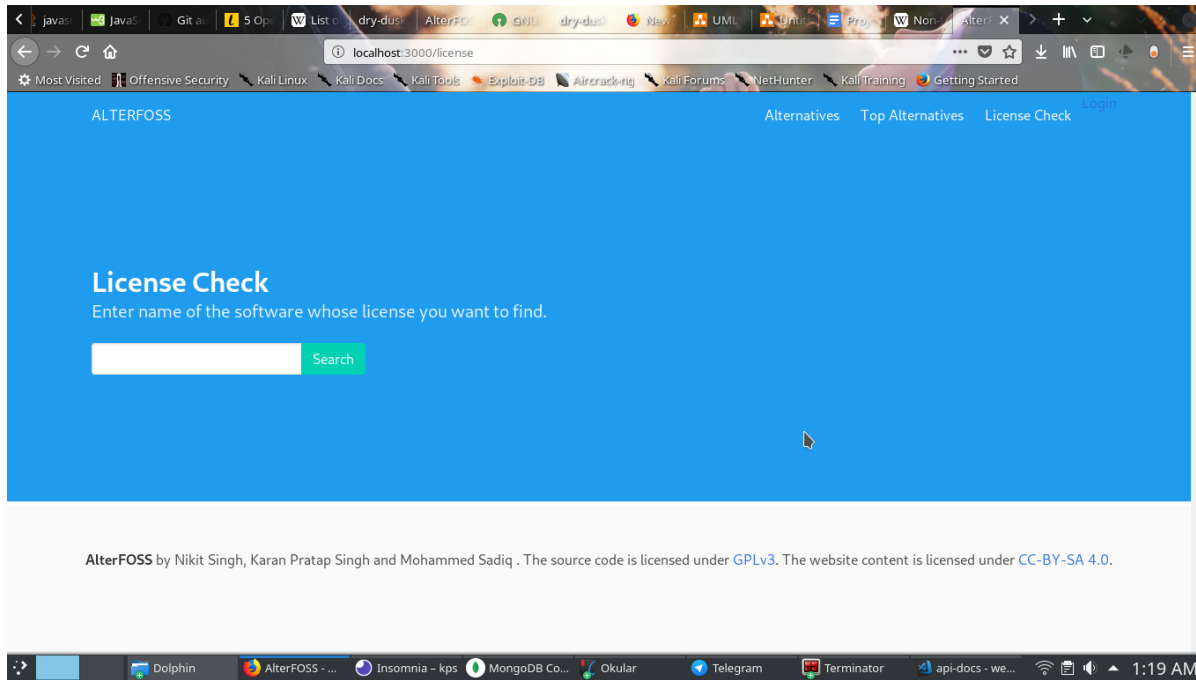


Figure 4.6: License Checker

The above depicted is the license checker module implemented in our mini project. The user can provide search text in the given text box to search alternative softwares. The result will be the name of the alternatives matching this search string along with their license information. Thus, the user can get the license information of any proprietary software he desires using this module.

The result of this query is as shown in the following figure (fig. 4.5), which shows the search result for the search query 'tor'. Since the items are matched with the search pattern using the regExp approach with case insensitive flag set, thus this search leads to displaying both TOR browser and GIMP photo editor.

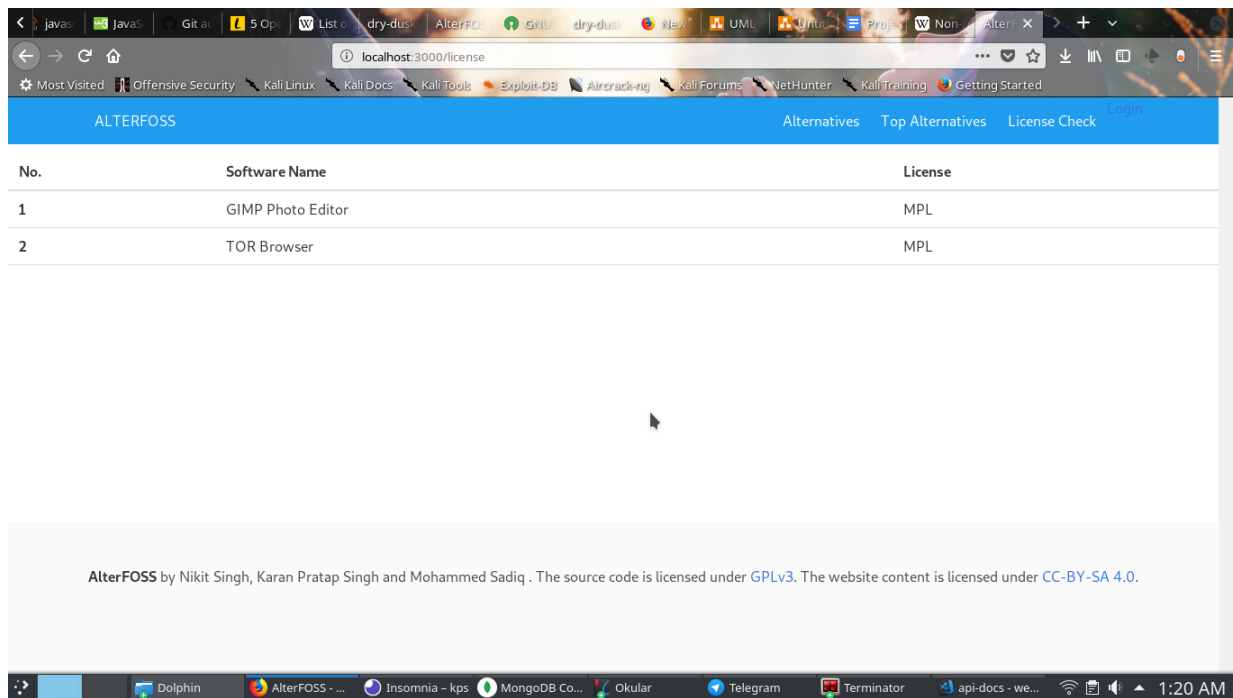


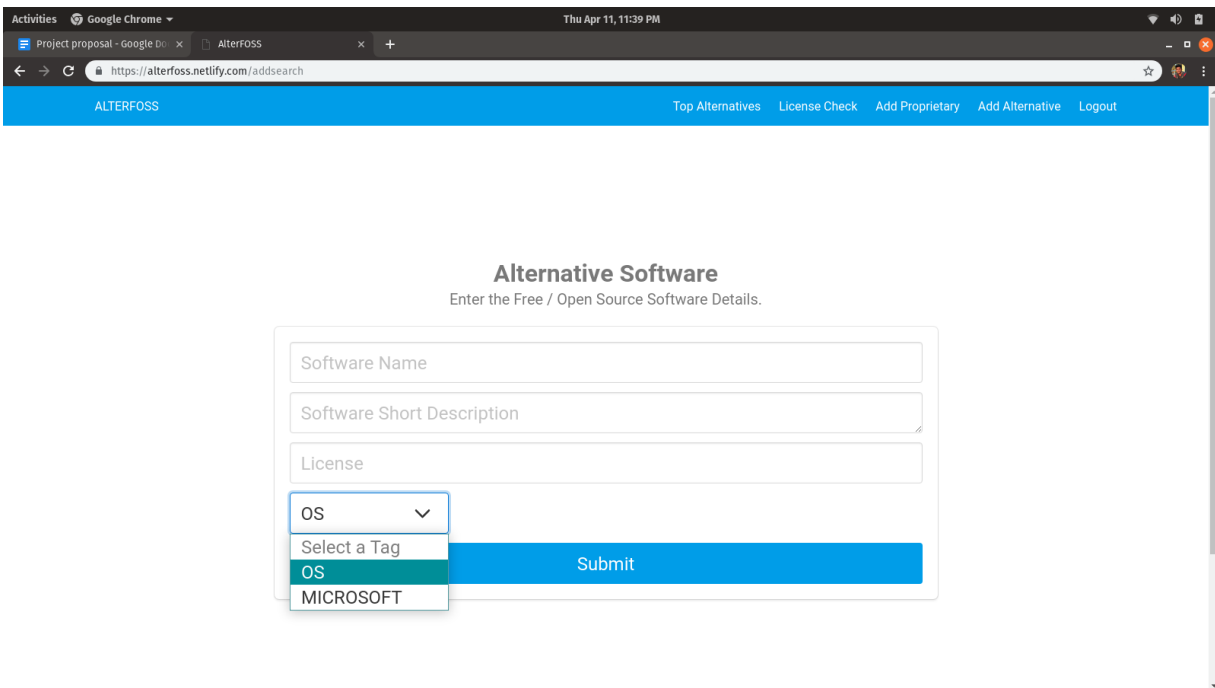
Figure 4.7: License Checker Output

The screenshot shows the ALTERFOSS Proprietary Software Detail form. The form has a blue header with the ALTERFOSS logo and navigation links: Alternatives, Top Alternatives, License Check, Add Proprietary, and Logout. The main heading is 'Proprietary Software Detail' with a subtitle 'Enter the Proprietary Software Details of which you want open source alternatives.' The form contains three input fields: 'FarCry3' in the first field, 'A rather very interesting game to play!' in the second field, and 'Game' in the third field. A blue 'Submit' button is located at the bottom of the form.

Figure 4.8: Adding a new proprietary software

As can be seen from the above figure, the user will have to provide the details of the software such as name, short description and tags via a form in order to request for this proprietary software's alternatives. The requested-by field will be set to anonymous automatically if the user is not logged in. Otherwise, the username will be fetched from the current session and will be assigned to the requested-by field. A json

object will be created using these details fetched from the user and this object is sent via the req.body property of the request. In response, the back-end will return the complete object written onto the database along with the `_id` and `_v` fields(though these can't be seen in the figure explicitly).



The screenshot shows a web browser window with the URL `https://alterfoss.netlify.com/addsearch`. The page has a blue header with the 'ALTERFOSS' logo and navigation links: 'Top Alternatives', 'License Check', 'Add Proprietary', 'Add Alternative', and 'Logout'. The main content area is titled 'Alternative Software' with the subtitle 'Enter the Free / Open Source Software Details.' Below this is a form with four input fields: 'Software Name', 'Software Short Description', 'License', and a dropdown menu for 'OS'. The 'OS' dropdown is open, showing options: 'OS' (selected), 'Select a Tag', 'OS', and 'MICROSOFT'. A blue 'Submit' button is located to the right of the form fields.

Figure 4.9: Creating a new alternative

Once the proprietary software is requested by someone as shown previously, the other community members will have the option to suggest alternatives for this proprietary software. This feature is also quite similar to that of the addition of proprietary software. The user needs to specify the name, license, short description and a handle for the alternative to be added. A json object is created using these properties and setting their values to those specified by the user via the front-end form. Here also, the suggested by field is set to the username from the current session, in case the user is logged in, otherwise the default value 'anonymous' is used. Both of these features to add an alternative and to create a proprietary software requires a 'POST' request to be sent to the server. These softwares are then stored inside the database and can be viewed by other users later. This way the platform will keep growing and the community will steer it.

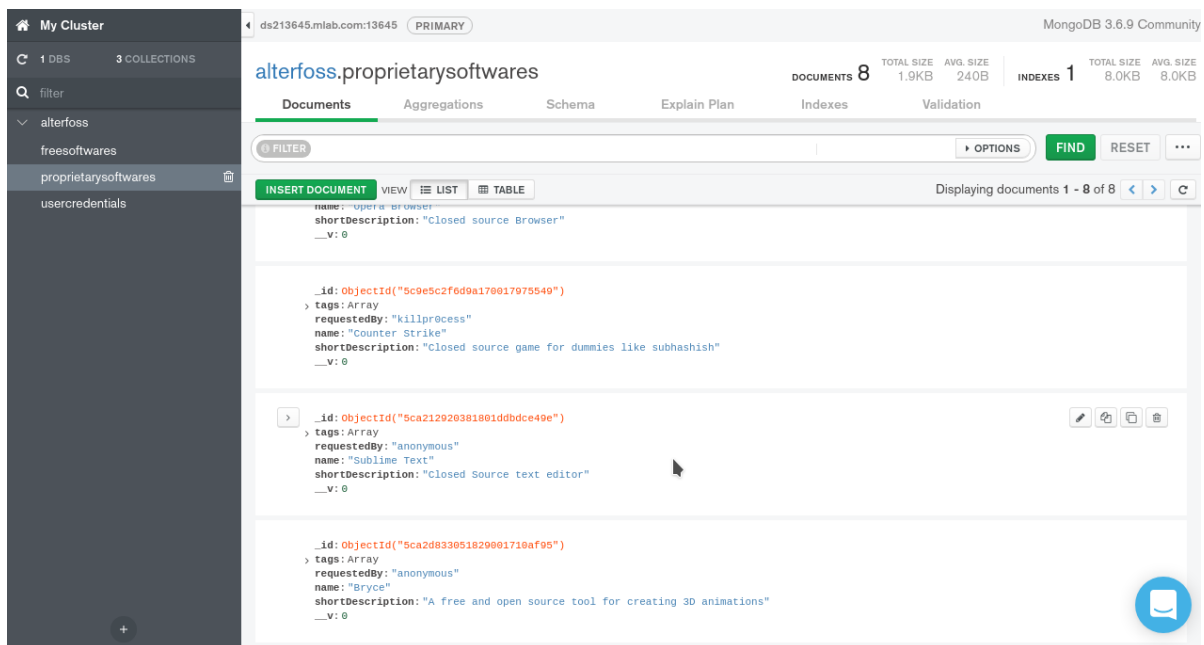


Figure 4.10: proprietarySoftware collections

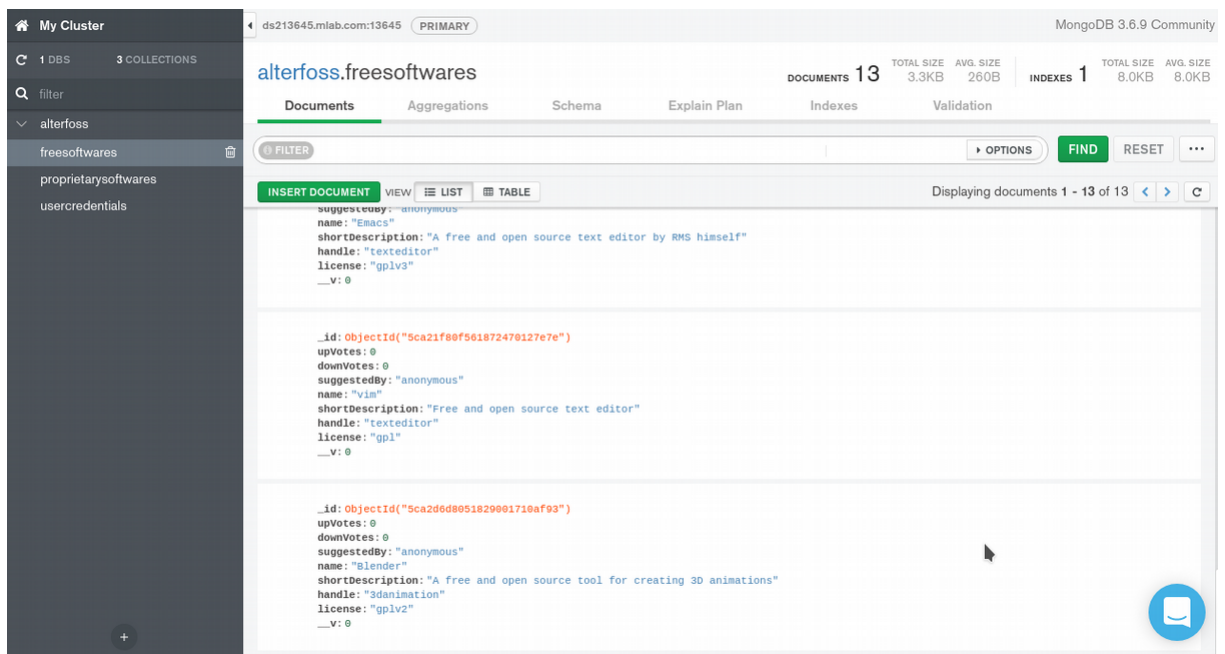


Figure 4.11: freeSoftwares collections

The above screenshots (fig. 4.10 and fig. 4.11) show a glimpse of the database after addition of these softwares.

Chapter 5

Conclusion And Future Scope

5.1 Conclusions

Free and open source softwares are important choices towards safeguarding ourselves as well as an appropriate preference for developers and users alike. Our project tries to do what little we can to make this world a better and more safe place. Applications like this reinforce the need and importance of FOSS as well as open an opportunity to experience life differently.

We really enjoyed doing this project and more importantly, learned so much along the journey that we are indebted to it. This project shall also find a place in the muse of FOSS evangelists and shall be taken forward as it finds its place in this dynamic world. We hope to see AlterFOSS grow further as well as bring about a moral conscience in all those who choose to use it and bring about a change in their lives. The next section shall talk about the future scope of this project and how we can improve on the shortcomings of present.

5.2 Future Scope

There were many things that we set out to do and there are many things among them that we did. But as always is the case, we still feel that there is a scope for improvement and we shall point out just that in this section.

First things first, so far we have restricted ourselves to suggest alternatives based on the software but it doesn't have to end there. We can suggest softwares based on the functionality they provide and based on the requirements specified. We can add comment sections and also an automated feedback section that would require a touch of intelligence to the whole system. We propose that we do logistic survey of our traffic without attributing it to individual users and start suggesting the softwares based on popularity and the feedback they receive. Comments are a great place to start wherein we can get an overall sense of the success of the given alternative and accordingly we can rate the software.

We further propose to add a pros and cons section for each software. Strictly speaking, we really wanted to pull this in in this version itself but due to time constraints, we would have to postpone it for the later versions.

As we come to the end of our report, we would want to end it by noting that there is far too much that can be done and we are open to suggestions and advice. As this project is very close to our ideals, we shall continue to provide support to it and bring about all that we discussed so far in a more optimized way and also, let the community make contributions too.