ask-rep

(Online repository)

Dhiraj Narwani (1431555), Ventsislav Polimenov (1457989)
Department of Computer Science, University of Bristol
BS8 1UB, UK

Abstract - **The main objective for this project was to create an online repository where users would be able to create, upload and share their files amongst other users. Users would be able to create public repositories in order to upload or create new files and store them in folders/sub-folders. The main reason for creating custom files is to allow the user to write pieces of code and if needs be whilst typing the application will recommend online code snippets to him/her. This will permit the user to search for code through the application instead of typing their query in Google.**

These files are saved on the cloud and placed in new or existing repositories according to the users' preference. All repositories are available to the public through trending repositories and the files can be viewed straight from the browser. An authentication system was implemented where guests can only access repositories created by users as well as the repository structure and files created in it. Alternatively, users can login into "ask-rep" and this would automatically connect to their Google account or request the necessary permissions needed. Moreover, the user would then be able to create personal repositories and customize them how they wish. The application can be run online at https://future-spot-815.appspot.com/.

Keywords—online repository, cloud, code snippets, files

I. INTRODUCTION

In the past few years we've seen a big leap in the demand for cloud computing. Google has been the prominent leader in creating robust, efficient and response applications specifically designed for public use. "ask-rep" is a cloud application which targets individual programmers and those employed in IT companies.

Our goal was to create a tool for the public to access their files from anywhere they wish. Instead of storing their information on a PC, the cloud allows them to access information from anywhere with fast response rates. This is highly useful for IT companies that have branches in different locations because the data can easily be accessible and modified from one central location.

"ask-rep" has specifically been designed to automatically scale up as more repositories, files and folders are created. As data and information grows, the users' experience improves because more data is available to be used. Googles' Cloud ensures scalability throughout the application and makes sure that performance remains efficient and responsive.

Generally as scalability increases, further profit is generated due to the increase in data and number of users accessing the website. In this scenario the application is free to use by anyone and doesn't include a payment scheme. Moreover, the intention was to provide a service to the public to store their information on the cloud free of charge rather than using third-party applications in which some of these need to be purchased.

In the web, users want to access quick tools which accommodates their needs. "ask-rep" introduces the custom search tool where developers will save huge amounts of time searching through Google looking for code snippets. Usually when developers write a piece of code, they're uncertain if they are going to get stuck at a specific point. Knowing that this tool is available to them in the background, ensures that they can use it whenever they wish. Ultimately this improves the user experience and raises awareness to other developers.

Security is one of the most important aspects in Cloud Computing. Nowadays, as IT companies and developers make use of the cloud, more hackers are trying to break in and tamper data. Since the web continuous to gets larger and is already highly exposed to everyone, cloud companies focus on providing the best security to stop hackers.

In "ask-rep" we enforced database security such as relationships between tables and the appropriate foreign key relations. Since some functions are restricted to users we implemented the necessary checks to prevent guests accessing them. We used session-based authentication to only allow users with an authorized session to create and upload files. Guests could also use the website by accessing repositories of other users and also viewing the structure and files within it. This allows anyone to use the website including those who don't wish to sign in or enroll with the website.

The client-server based model allowed us to cater for failed and success responses. This prevents the website from crashing due to random exceptions occurring. In turn, by means of saving user information into the database, it can allow us to track unwanted users from spamming or harming the website intentionally.

We see that "ask-rep" is mainly dependent on user-generated content. As users upload their information, they are shaping and building the application. Primarily by sharing their information with others allows them to continue build the application faster as well as shaping it by including a huge amount of information in a short period of time. This minimizes the amount of maintenance required so the application can solely function on user-generated content.

Another benefit that Google Cloud provides us with is that the application has low latency and reliable network connections. By increasing the number of users from different locations the application still remains robust and efficient. This spread out enables the website to expand with information and user-generated content. Google ensures their servers cater for users from all locations around the world. The appropriate servers are used depending on where the application is being accessed from. This results into faster speeds due to minimum proximity.

II. PLANNING

The project took 57 days to finish (see Fig. 1) and tasks were divided as follows:

Dhiraj Narwani

- Front-end design and implementation (HTML, CSS, JavaScript and Java code).
- Database Creation – All files, repositories, folders, users will be stored in a database.
- Sign in/out functionality – Users can login with their Google account to access restricted functionalities.
- Customizing repositories – Creating a repository where folders and sub-folders can be included.
- Create files – Save files in the database that are created by the user through a text editor.
- Upload files – Upload existing code files with specific extensions to the database.
- Viewing files – View existing files from public repositories in the browser.

Ventsislav Polimenov

- Google Custom Search – Functionality that will query google in order to extract code tags.
- Extract code tags – Extracting code snippets through ‘code tags’ from pre-defined websites.
- Display code – Displaying code snippets in panels.
- Trending Repositories – List of all repositories in descending order (recent first)

III. IMPLEMENTATION

The project was coded in Java using Google's App Engine API. This uses the client-server framework where the back-end code is created on the server and interfaces are used to access the server code. The client interfaces are used through an “entry point” which translates Java/object-oriented code into JavaScript onto the front-end. This ensures efficient responses and minimum delays for users.

We used two toolkits provided by Google (GWT [1] and SmartGWT [2]) which are open-source software tools that help developers better manage the client-server framework. Google Web Tools (GWT) provided us with GUI components to create a structured application design through our “entry point”. This included textboxes, html placeholders, panels, buttons etc…

A. Front-end design

We started off by creating the front-end design including HTML and CSS for our cloud application. This included the website theme, menu structure, logo and custom buttons. A database was needed to manage and store data such as user information.

B. Database and instance configuration

Initially we created the database locally using MySQL and created the appropriate table structure. Our database consisted of 4 tables; users, repositories, folders and files. This enabled us to create a repository-like structure to manage folders/sub-folders and files. We ensured relationships between tables were defined to keep the database secure. In order to deploy it onto the cloud we used Google Cloud SQL service.

We signed up with Google Cloud platform and automatically got assigned a personal console management. Through this console we created a project and added the Cloud SQL service. We created an instance which Google provided us with an IP address linking to the SQL database. This enabled us to connect to Cloud SQL and import the local database through command line.

We chose Cloud SQL rather than Cloud Datastore or Storage because it allowed us to store user information linked with repositories/folders and files. Therefore, the application can display the repositories of the current user logged in. The other two alternatives only stored folders and files, and it’s not ideal of creating a repository-like application.

C. Login

In order to access specific application functions such as uploading and creating files, users need to sign in using their Google account. The user information is stored in the database which allows us to keep track of the files/folders/repositories they create as well as a timestamp for each record created. Also, this permits users in having their own personal repositories in a self-managed manner.

Users who are not linked with a Google account or don't wish to sign in can browse through Trending Repositories created by other users. We will discuss this in Section IV.G Trending Repositories.

D. Create/Edit Repository

Once a user logs into the application he/she can create/edit personal repositories. In the menu there are two links; “Create Files” and “Upload Files”. The application will show the user a section where he/she could create or select an existing repository. This allows the user to create fresh repositories or edit an existing repository they created previously. This data is stored in the repository table along with created and updated timestamps.

E. Customize Repository

In the repository section users can add folders and files to organize their data properly. They can easily navigate through existing folders in the repository by clicking on a respectful folder name. This provides the user with a repository experience where he/she could create infinite folders and easily traverse back to the root.

Alternatively they can add custom folders by clicking on “Create Folder” located next to the directory listing. The user can see the path of which folders they've accessed and the application will indicate the user the one he/she is currently accessing. The created folder will then be placed in the current directory.

The database structure permits the application to store folders in a single table where each folder links to a parent folder unless it's created in the root. Once a folder is created, the timestamps are updated for its parents and its linked repository. This provides the user with continuous updated data where he/she can keep track of the folders/repositories that were last accessed.

F. Create/Upload/View Files

Whilst accessing a specific repository the user can be able to create a file and write their own piece of code in the application. Instead of using an external application such as “eclipse” they can create code snippets or files and store them in their own repository on the cloud. The user is provided with a code panel (text area) which includes customized tabbing so he/she can indent their code like “eclipse”.

Furthermore, if the user wishes to upload external files, he may do so by uploading single or multiple files in specific repositories. The applications restricts the user to 5 extensions (code languages) which are “.py, .js, .cs, .cpp and .java”. We used a third party library called “GWT Uploader” [3] which allows the user to upload multiple files. It enables you to handle and use specific events in turn improving the user experience.

The files in each repository are listed accordingly to where the user has created them. The user can view the files' contents by clicking on the respectful file name. They won't be able to download the file to their PC but only view it through the browser. This also applies to guest users who don't sign into the application because they can view files created by other users.

G. Trending Repositories

As you enter the application, immediately it will list you all the repositories that have recently been created. They are sorted in descending order by repository date. Guests will be able to browse through recently created repositories and view their files and folders. This provides users/guests with constantly updated information depending on how much the application is being is used.

Alternatively an authenticated user can view all repositories that are created only by other authenticated users. Since they can already view their own repositories whilst creating/uploading files we decided to remove the duplications and suggest better repositories to them.

H. Search functionality

We implemented a functionality where users can be able to search for code snippets through the internet using Google Custom Search Engine. Whilst users are creating files, if they get stuck and don't know how to write a specific function, they can send a request to Google and retrieve the code snippet.

- *Hot-Button for quick search*

We provided the user with the choice of 5 languages – JavaScript, Java, Python, C++, and C#. When creating files the user needs to select the language from a dropdown list and insert a filename. The user can select the world/phrase and by pressing the button SHIFT a query will be sent to Google.

A website was needed in order for Google to crawl through it an extract code snippets; we chose “http://www.stackoverflow.com”. The end of the query consists of the user's highlighted phrase and chosen language followed by the name of the website.

- *Custom Search Engine*

This is an API provided by Google, so that developers can integrate customizable search engines in each and every website as well as in online applications. To integrate the API we used the search engine' URL (placed at “googleapis.com/customsearch/”), the cx_key, which serves as an authenticator for the engine, the API_KEY provided by Google which acts as credentials to the cloud application, and a parameter that enforces JSON output format.

- *Jsoup Library*

When the user's query is sent to Google, our application gathers all the results as links to different snippet web pages and stores them in an Array List. Once this process is done, the same data structure is sent to an iterator, which opens each link in order to access the html code for every web page. The iterator seeks for “<code>” tags in the HTML and when found extracts the text within and inserts it into a HashMap. Therefore, for this functionality we used an external library for HTML stripping called Jsoup [4].

- *Code Snippet Rating*

In order to achieve plausible results for the ultimate user experience, we implemented a rating method which rates the extracted code snippets before they actually get displayed for the user. Firstly, we have predefined language code tags that serve as identifiers for each language. If the language of the extracted snippet does not match the language chosen by the user, snippet gets penalized. However, if the searchable keyword occurs more than once the rating increases by 5 points.

- *Displaying Code Snippets*

For displaying the snippets we decided to use an “accordion” type of container that manages a list of sections of widgets, each with a header provided by SmartGWT [5]. We gather a minimum of twenty code snippets and display them next to the text area assigned for coding. Also, for each code snippet we displayed the URL of its source.

IV. Deployment & Results

After implementing and testing the application locally, we uploaded it onto Google Cloud platform. We deployed the application through "eclipse" by linking the application id – "future-spot-815" with the source code. Currently Google provides us with 60 days free trial so we didn't need to choose a pricing plan for hosting. Once the trial expires, we would need to think about choosing the right plan to host the application depending if our intention is to keep it live or not.

The end product resulted into a fully pledged cloud application. Users are able to create repositories with a proper repository structure, create or upload code files and also seek for code using Google Custom Search Engine. Google provides the code snippets and our rate algorithm chooses the best ones.

V. Future Work

Regarding future improvements, we could improve our text editor to detect tokens and highlight syntax. Also we can integrate to it both run/compile and debug functionalities so that it would not be just a simple text editor, but an online IDE. This improvement is very difficult to implement but, perhaps, it could allow us to make money off the application.

Another improvement we could implement is to enhance our rating algorithm, which is based on language classification and word occurrences. As mentioned in Section H. the current language classification is based on a hard coded array of most commonly found words for each language. However, we could use a more sophisticated learning technique, which can detect and understand the programming language entirely on its own and use better ranked words according to the language. We can implement this using Machine Learning and possibly might suggest better and more accurate code snippets to the user. This could possibly increase user efficiency and complexity of code functions.

A smart and profitable future enhancement could, perhaps, be to integrate Google AdWords. We can integrate ad-click and for each advertisement that is clicked on, we get a share from Google's profit off advertisers. Our biggest expense would be to pay for hosting but if we manage to get a good number of users accessing the website then we could make a reasonable amount of profit.

The advertisements could be related to programming or specifically to repository descriptions. Users that are accessing a repository might be interested in certain functions so there's a high possibility of them clicking on an ad related to that specific repository. These enhancements all involve a great deal of time but most importantly is that we managed to come up with an idea that can easily grow and keep on expanding.

VI. Conclusion

In this report we gave a detailed discussion about all the resources and tools that we used in order to develop a Cloud application on Google Cloud platform. This includes all the functionalities of the application and how they can be used. A user specifically a programmer can use this repository for personal use. He/she can store files and folders to manage his/her projects efficiently. By doing so, they can easily create code files and avoid time wasting just by using our search function instead of visiting Google. Since individual programmers and IT companies are both pressed for time when implementing projects and keeping up with deadlines, this can serve them a great deal of time.

As an overall, we are definitely satisfied with the outcome of the project as we were able to implement all the functionalities we were aiming in the designated timeframe. The current state of the product allows us to put it live on Google Cloud and most probably continue to expand with profitable intentions.

VII. References

[1] "[GWT]" Accessed 25 Dec. 2014: http://www.gwtproject.org/

[2] "smartgwt – Smart GWT – GWT API's for SmartClient – Google Project Hosting" (2014) Accessed 26 Dec 2014: https://code.google.com/p/smartgwt/

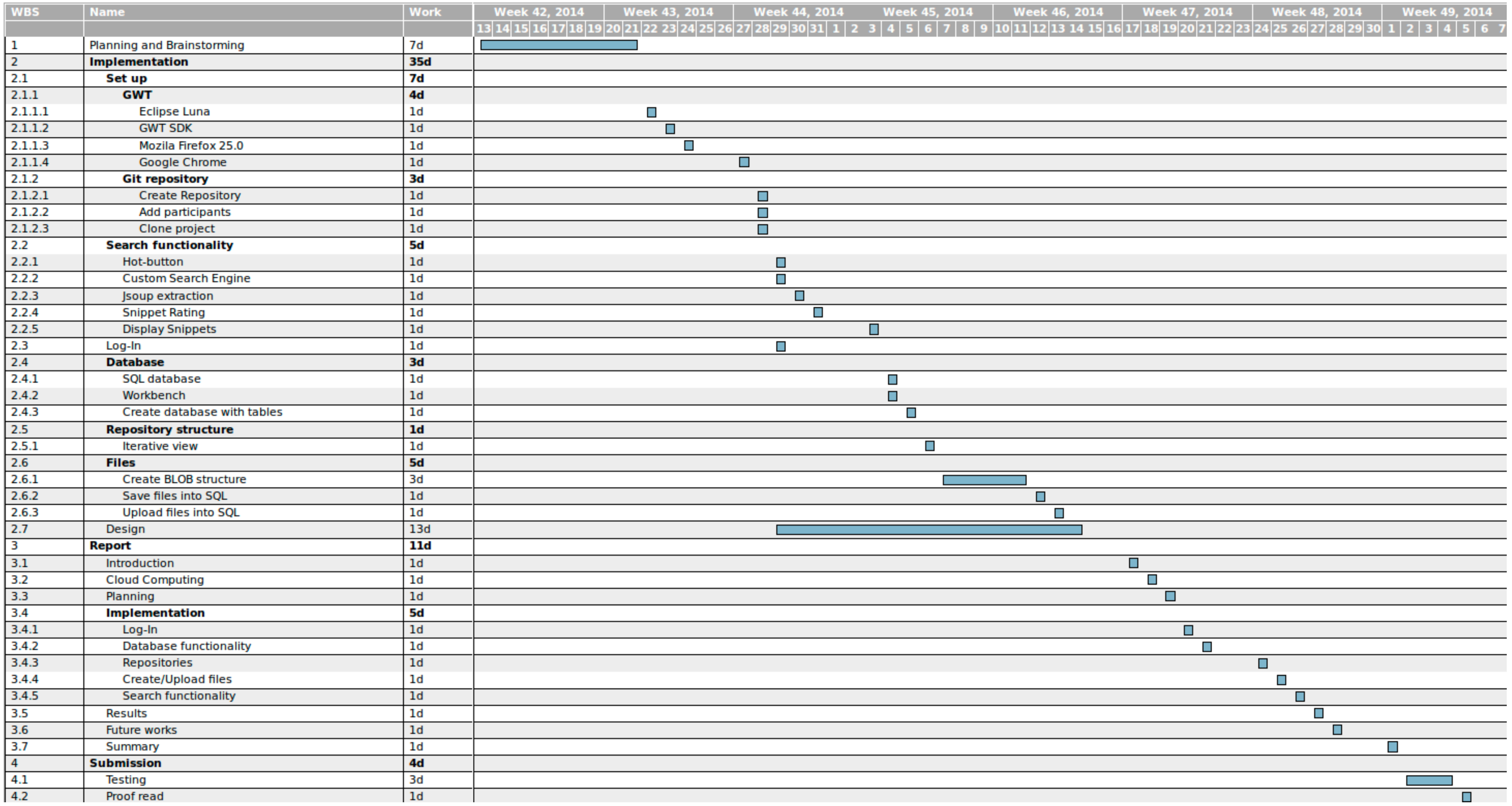
[3] "Moxie Group: GWT Uploader – Overview" Accessed 28 Dec. 2014: http://www.moxiegroup.com/moxieapps/gwt-uploader/

[4] Jonathan Hedley. "jsoup Java HTML Parser, with best of DOM, CSS and jquery" (2014) Accessed 28 Dec. 2014: http://jsoup.org/

[5] "Section Stack" Accessed 30 Dec 2014: http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/widgets/layout/SectionStack.html

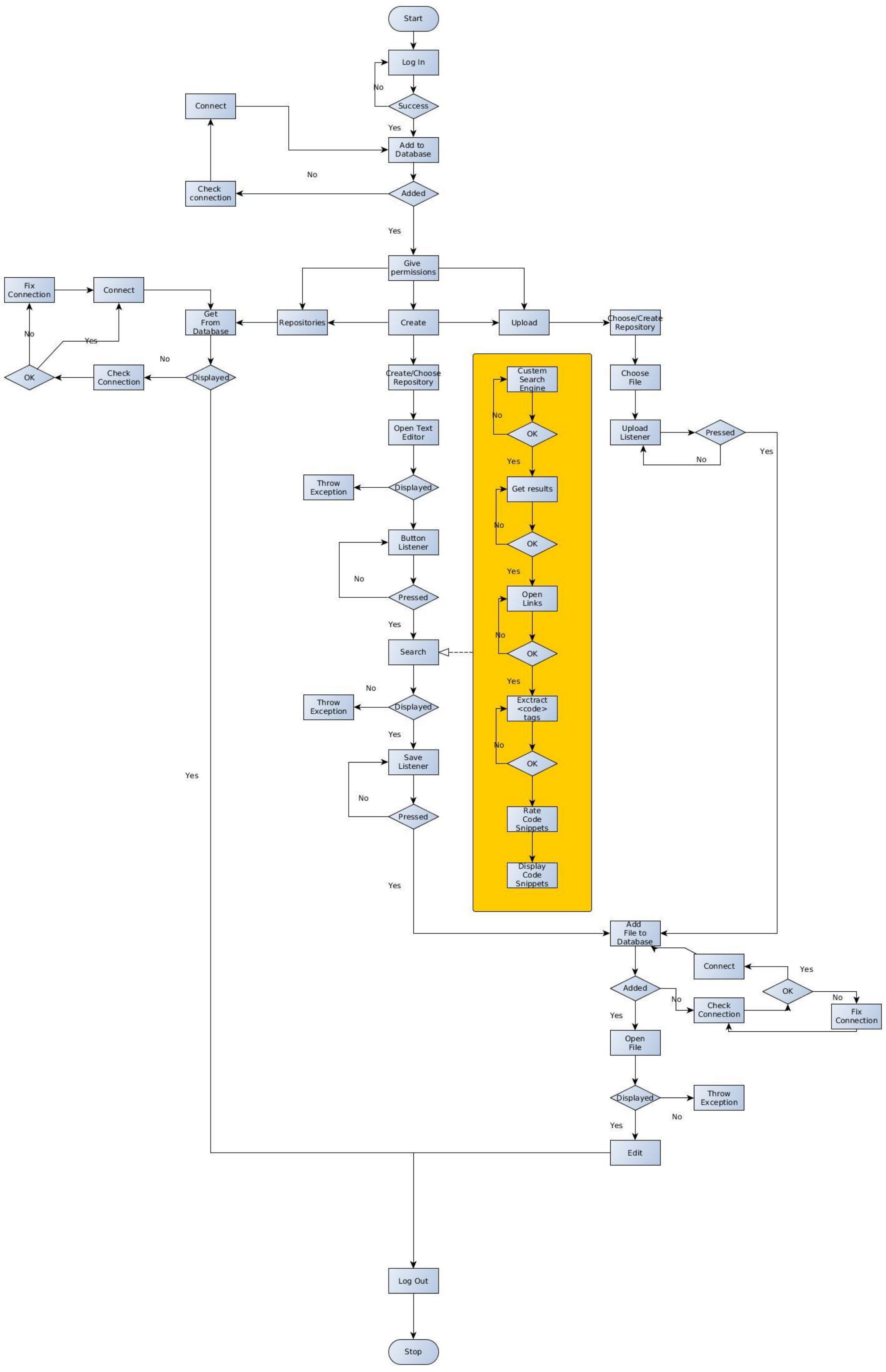
Fig 1. Gantt chart with entire plan from beginning to final end date of Cloud application: “ask-rep”

Gantt Chart



WBS	Name	Work
1	Planning and Brainstorming	7d
2	**Implementation**	**35d**
2.1	**Set up**	**7d**
2.1.1	**GWT**	**4d**
2.1.1.1	Eclipse Luna	1d
2.1.1.2	GWT SDK	1d
2.1.1.3	Mozila Firefox 25.0	1d
2.1.1.4	Google Chrome	1d
2.1.2	**Git repository**	**3d**
2.1.2.1	Create Repository	1d
2.1.2.2	Add participants	1d
2.1.2.3	Clone project	1d
2.2	**Search functionality**	**5d**
2.2.1	Hot-button	1d
2.2.2	Custom Search Engine	1d
2.2.3	Jsoup extraction	1d
2.2.4	Snippet Rating	1d
2.2.5	Display Snippets	1d
2.3	Log-In	1d
2.4	**Database**	**3d**
2.4.1	SQL database	1d
2.4.2	Workbench	1d
2.4.3	Create database with tables	1d
2.5	**Repository structure**	**1d**
2.5.1	Iterative view	1d
2.6	**Files**	**5d**
2.6.1	Create BLOB structure	3d
2.6.2	Save files into SQL	1d
2.6.3	Upload files into SQL	1d
2.7	Design	13d
3	**Report**	**11d**
3.1	Introduction	1d
3.2	Cloud Computing	1d
3.3	Planning	1d
3.4	**Implementation**	**5d**
3.4.1	Log-In	1d
3.4.2	Database functionality	1d
3.4.3	Repositories	1d
3.4.4	Create/Upload files	1d
3.4.5	Search functionality	1d
3.5	Results	1d
3.6	Future works	1d
3.7	Summary	1d
4	**Submission**	**4d**
4.1	Testing	3d
4.2	Proof read	1d

Fig 2. Flowchart describing process for repository functions mentioned in Section IV. Implementation. The yellow box shows the process for the Google Custom Search functionality


Start
Log In
Success
No
Yes
Add to Database
Added
Connect
Check connection
No
Yes
Give permissions
Repositories
Create
Upload
Fix Connection
Connect
Get From Database
Displayed
Check Connection
OK
No
Yes
No
Create/Choose Repository
Open Text Editor
Displayed
Throw Exception
Button Listener
Pressed
No
Yes
Search
Displayed
Throw Exception
No
Yes
Save Listener
Pressed
No
Yes
Custom Search Engine
OK
Get results
OK
Open Links
OK
Exctract <code> tags
OK
Rate Code Snippets
Display Code Snippets
Choose/Create Repository
Choose File
Upload Listener
Pressed
No
Yes
Add File to Database
Connect
Added
Check Connection
OK
Yes
No
Fix Connection
Open File
Displayed
Throw Exception
Edit
Log Out
Stop