Online-Learning in Social Networks
— Report Two —

Shanshan Fu, Oh Boon Eng, Ani Petrosyan, Magdalena Sadowska, Shahrokh Shahi, Yunxin Zhou
{sf2715,beo12,ap7614,ms915,ss6515,yz5115}@ic.ac.uk

Supervisor: Dr. Alessandra Russo
Course: CO530/533, Imperial College London

11th May, 2016

Contents

1 Introduction

Today, with around 3 billion people using some form of the Online Social Network (OSN)[1], the difficult challenge of preserving one's privacy whilst still fulfilling their social needs is almost inevitable. In order to decrease such difficulty, OSN platforms often allow the uploaders to define a target audience of the information they post. However, some of the members of the preselected group may choose to re-share the information, which could lead to unexpected privacy breaches[2]. As a result, the uploaders might not always anticipate the negative consequences of their online activities and engage in actions that they may later regret[3, 4].

The aim of this project is to develop a web plugin based on the research conducted by Imperial College London. The study proposes a Learning to Share product, in which a parametric Markov chain (PMC), a chain with unknown transition probabilities, is utilised to evaluate the privacy risk and social benefit of sharing an information with each friend or a member of a predefined target audience. The main functionality of the web plugin is to be able to identify potential malicious friends based on previous interactions between the user and his/her friends and return an appropriate warning message to the user when the user wants to share a post with said friends so as to prevent the user from sharing information that may be leaked to unintended audiences.

The approach models the interactions between the user and each group member as a discrete-time Markov chain (DTMC) with parameterised transition probabilities. These probabilities are learnt and continually updated at run-time, based on the actual interactions between the user and the group member. Whenever the user considers sharing a new message with a group member, the parametric DTMC is formally verified to establish whether the member still meets the user's sharing requirements. These requirements specify the maximum level of privacy risk and the minimum level of social benefit that he user is willing to accept in order to share the new message with the group member. The requirements take into account the sensitivity level of the message, and are formally expressed in probabilistic temporal logic.

Being currently the most widely used OSN that provides a variety of user interactions[5, 6], Facebook will be considered as the prominent case study in this project.

1.1 Background

1.2 Learning to Share Approach

1.3 Facebook Terminology

Facebook has many features. Within this project, those components that are common in most of the social networks are of interests. To establish a tool relating the user's privacy, acquaintance with these features is inevitable. The crucial components of Facebook are as follows:

Timeline: The Timeline can be considered as the main feature of the Facebook in which all the contents of the user's profile, including posts, photos, videos, etc., are categorized and depicted according to the period of time in which they are posted or created. The Timeline is actually a new version of the Wall that was the regular space of the user's content.

Like: Like is a way to give positive feedback to contents, e.g. posts, comments, photos, etc. Recently, Facebook launched its new ”reaction” buttons, giving users the option of using ”reactions” such as ”Love” ”haha” ”Wow” ”Sad” and ”Angry” in stead of Like.

Notifications: Notifications tell the user that something has been updated in his or her profile page or in a friend's status, if he or she has been specified as a close friend. These updates can be a comment on a picture of the user, an answer to a user's comment, a message posted on user's timeline, etc. The user will be notified by a notification counter in red at the top of the page, which if clicked displays the most recent ones.

Friends: The fundamental functionality of a social network is friending that is adding someone to a list of "friends" who the user wants to have interaction with. The act of friending someone grants the privilege of viewing and posting on the user's timeline.

News Feed: This component enables user's to track of the others' activities, such as new posts, photos, life events.

1.4 Report Structure

This report describes the development process of the plugin. It is divided into the following sections:

- **Specification:** The requirements of the each component of the plugin, broken down into small tasks for the members of the group to handle.
- **Design:** Detailed description of the web plugin and the functionalities of each of its components.
- **Methodology:** Development technique adopted by the group, issues faced during the development process as well as testing methodology.
- **Group Work:** Assignment of tasks to each group member
- **Final Product:**

2 Specification

ADD SOME STUFF HHERE

2.1 Requirements

Main high level essential requirements of the web plugin proposed by the research group include the creation of a component, that is responsible for monitoring the social interaction between the user and each friend, a user interface that allows the user to specify his/her settings, as well as adding an automatic manager responsible for recommendation to the user to select a proper audience group. Table 1 demonstrates the low level specifications derived from the aforementioned requirements by applying the simplified version of van Lamsweerde's KAOS goal-oriented model [7], questioning how and why a particular requirement is applied. Table 1 is grouped by a particular component of the plugin and includes a relevant risk levels.

Table 1: Specifications

	Component	Description of Specifications	Risk*
Minimum Specifications	Facebook API	Creating an interface to get permission from user through facebook API	L
		Creating a user interface that allows the user to input the level of sensitivity as well as the privacy risk and social benefit thresholds	L
	Database	Creating a database in form of a hash table to store information and sensitivity level	L
	Monitor Module	Keeping track of interactions between the user and selected friends	H
	Parametric Model Checker	Integrating of existing parametric Markov chain model with the application	L
	Online Parameter Learning Engine	Integrating of existing code that updates the parameters of the parametric Markov chain model	L
	Adaptive Sharing Analyzer	Creating a privacy risk and social benefit calculator	L
		Verifing if the user specified sharing preferences are met	L
		Calculating utility trade-off between social benefit and privacy risk	H
		Identifying the potential malicious friends and report them to the user	L
Possible Extensions	Database	Development of a database in place of hash table storing relevant data	L
	Online Learning Model	Automate the generation of the online learning model for any online social network based on the activity diagram on the network	H
	Level of Sensitivity	Explore machine learning methods and algorithms to improve determination of level of sensitivity	H
	Parametric Model Checker	Automate producing the algebraic expression through PRISM for other social networks	H
	Plugin Support	Extension of the plugin to support multiple browsers and exploring the possibility of support on mobile devices	H

*Risk index is categorised as high (H) or low(R), depending on its impact and occurrence. The following were considered in the classification: human, i.e. lack of development experience, software, being bugs with chosen tools and frameworks, and schedule, meaning inability to complete given specifications within given time framework.

Additionally, the non-functional specifications that need to be taken into consideration are performance and security. As far as performance is concerned, the application of concurrency will be implemented. The security, on the other hand, will be met by a strict implementation of Facebook API, as well as a thorough testing of the plugin.

Initially, the essential requirements of the web plugin were concerned with monitoring social interactions within a user's post in a group. However, after application of the said KAOS goal model, it became obvious that monitoring of the user's post in a group would not be feasible. More specifically, during the research concerning how to implement such an aspect, it was discovered that selecting the target audience of shared information is not allowed, therefore making it impossible for the web plugin to enable the exclusion of malicious members. That discovery led to a significant change in requirements from the group setting to monitoring of the Facebook Timeline activities, which introduced a new probabilistic model, as described in Section 2.

2.2 Updates to the original specification

The Table 1 summarises the progress of the project to date.

Table 2: Minimum requirements

Component	Tasks	ECD*	Status
User Input Plugin	Creating a log-in section	Phase 1	Completed
	Getting access token from Facebook	Phase 1	Completed
	Creating settings section	Phase 1	Completed
	Creating warning message section	Phase 1	Completed
	Creating server to pass and obtain information between plugin and monitor	Phase 1	In progess
Monitor Module	Getting user's friends list	Phase 1	Completed
	Accessing user's notifications to track the interactions	Phase 1	Completed
	Fetching all of the user's posts, related comments and likes	Phase 1	Completed
	Calculating the transition probabilities for Parametric Model Checker	Phase 1	Completed
	Integrating the monitor with the other parts	Phase 1	Completed
Parametric Model Checker	Refactoring the code to work with the real time data	Phase 1	Completed
	Integrating the code with the plug-in	Phase 1	Held up
Online Parameter Learning Engine	Integration of existing code that updates the parameters of the parametric Markov chain model	Phase 1	In progess
Adaptive Sharing Analyser	Privacy risk calculator	Phase 1	Completed
	Social benefit calculator	Phase 1	Completed
	Utility trade-off calculator	Phase 1	Completed
	Creating a method to compare the outcomes with user specified preference	Phase 1	Held up

*ECD - Expected Completion Date

Although significant progress with regard to the development of the plugin has been made, two essential requirements were marked as held up. Also, the current stage of the plugin development can be evaluated to Phase Two as opposed to initially estimated Phase Three. The reasoning behind the delay is a number of limitations encountered during the development, as presented in the following sections (2.1 and 2.2).

2.3 Specifications Update

The table below provides the updates to the specifications described in Table 1.

Table 3: Updated Specifications

Component	Description	ECD
User Login	Submitting a review to Facebook to obtain required permissions	Phase 2
User Input Plugin	Developing messaging system to communicate between two types of plugins	Phase 2
	Researching whether accessing Facebook's DOM is a practically feasible solution	Phase 2
	Adding terms and conditions and user agreement page	Phase 2
	Exploring an alternative construction of the plugin consisting of only one type, i.e. the popup	Phase 2
Monitor Module	Coming up with a reliable way to read the user's notifications, making sure none of the interations are missed	Phase 2
Parametric Model Checker	Updating the model to include Facebook's new feature (reactions)	Phase 2

2.4 Schedule updates

The major update to the schedule includes extension of the Phase One from 13th to 27th of March due to reasons listed in Section 2.1 and 2.2. The final deadline applies to all of the components, as each is being developed concurrently by each member of the team. Moreover, due to extension of the Phase One, the remaining Phases had to be shortened.

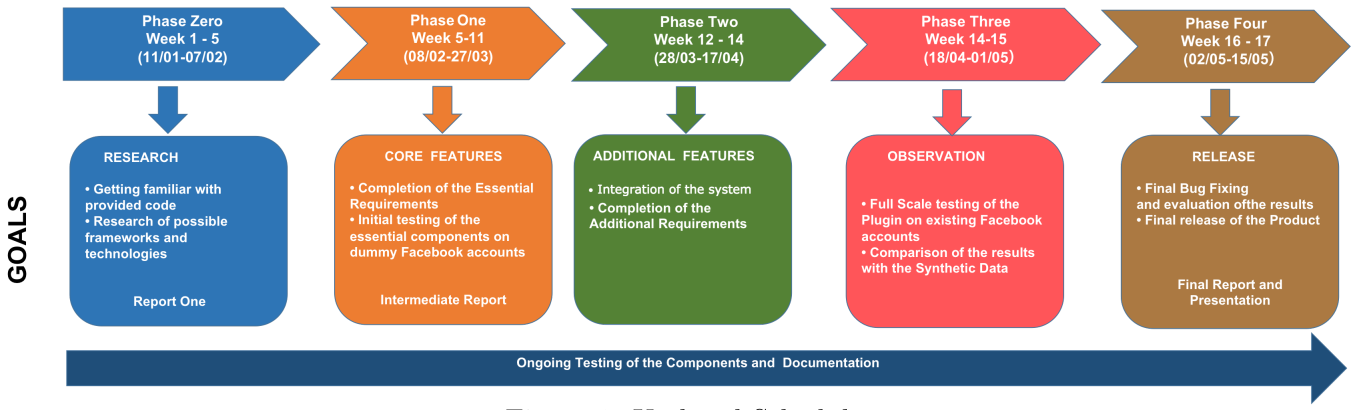

Phase Zero
Week 1 - 5
(11/01-07/02)
Phase One
Week 5-11
(08/02-27/03)
Phase Two
Week 12 - 14
(28/03-17/04)
Phase Three
Week 14-15
(18/04-01/05)
Phase Four
Week 16 - 17
(02/05-15/05)
GOALS
RESEARCH
• Getting familiar with provided code
• Research of possible frameworks and technologies
Report One
CORE FEATURES
• Completion of the Essential Requirements
• Initial testing of the essential components on dummy Facebook accounts
Intermediate Report
ADDITIONAL FEATURES
• Integration of the system
• Completion of the Additional Requirements
OBSERVATION
• Full Scale testing of the Plugin on existing Facebook accounts
• Comparison of the results with the Synthetic Data
RELEASE
• Final Bug Fixing and evaluation ofthe results
• Final release of the Product
Final Report and Presentation
Ongoing Testing of the Components and Documentation

Figure 1: Updated Schedule

2.4.1 Removed

- **Adjusting the learning model to fit Facebook's new "reactions":** This task proved to be too challenging as the whole learning model had to be changed. Furthermore, there are many more states to account for and more factors to consider. For example, if someone expresses anger towards a user's post, it is not considered a privacy risk as no information is leaked, however, it is also not a social benefit as anger is not a positive emotion.

- **Obtaining permissions from Facebook to request the user for permissions to use their information:** Facebook has been extremely strict with regard to allowing use of their permissions. After repeated submissions, it was decided that obtaining permissions was not feasible.

3 Design

The proposed Learn to Share approach consists of two fundamental aspects, being social benefit and privacy risk of sharing information. Social benefits is represented by positive response gained through sharing of the information, whereas the privacy risk is defined as the risk of being exposed to sharing sensitive information with an improper audience.

In order to evaluate the said privacy risk and social benefit, the research group proposes a web plugin that dynamically monitors interactions between the user and group members, categorising them into one of the predefined groups based on their potential privacy violation and displays an appropriate warning message, if necessary.

3.1 System Architecture

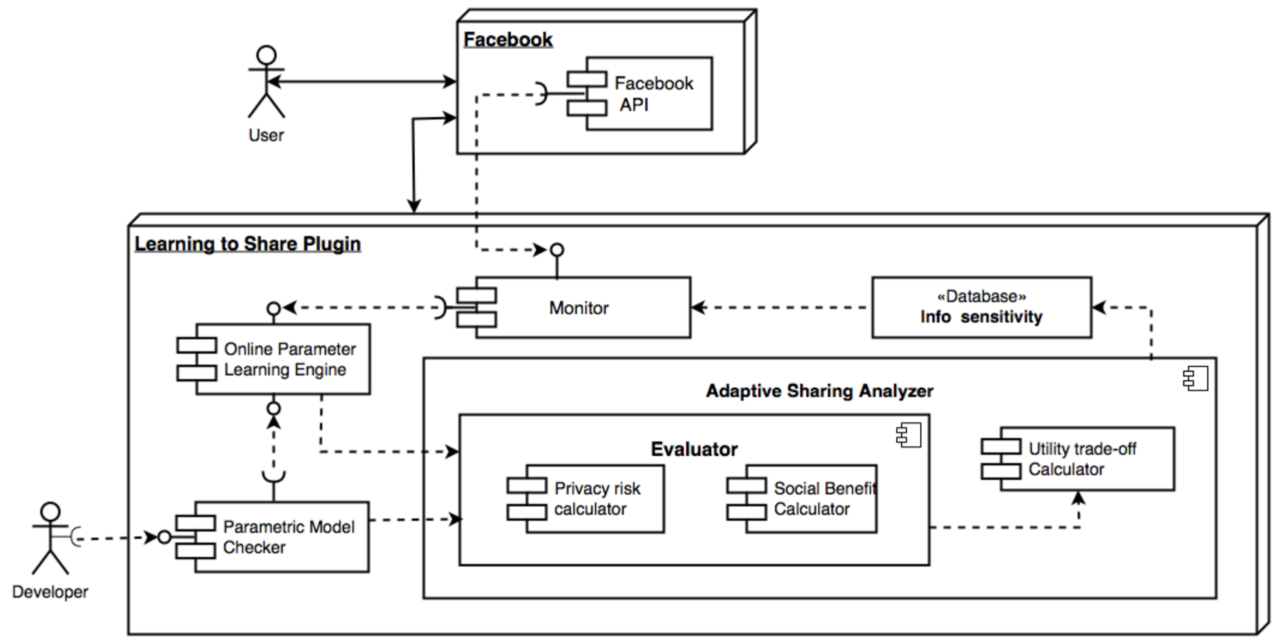

Facebook
Facebook API
User
Learning to Share Plugin
Monitor
«Database» Info sensitivity
Online Parameter Learning Engine
Adaptive Sharing Analyzer
Evaluator
Privacy risk calculator
Social Benefit Calculator
Utility trade-off Calculator
Parametric Model Checker
Developer

Figure 2: Component Diagram

Consequently, the most essential components that will make up the web plugin are as follows (Figure 1).

- **Facebook API:** The user logs in into Facebook and inputs the information along with the target audience. The API also allows the user to specify the social benefit and privacy risk thresholds, as well as the level of sensitivity. Besides, each piece of information, together with its corresponding level of sensitivity, is stored in the database.
- **Parametric Model Checker:** Takes two essential parameters as inputs: a parametric Markov chain model and the user's sharing preference for the social benefit requirement, which can be obtained through probabilistic calculation tree logic (PCTL) formula. The latter one is an algebraic expression produced using an open-source probabilistic model checker, PRISM.
- **Monitor:** Keeps track of the interactions on the user's posts, which will be used in the learning model.
- **Online Parameter Learning Engine:** Updates the model parameters
- **Adaptive Sharing Analyzer:** Uses the updated parameters that are used to calculate the privacy risk and social benefit. The calculated values are then compared with the user- specified thresholds to estimate the trade-off, which is then used to determine if the information is safe to be shared with the selected audience.

3.2 Program Flow

Furthermore, Figure 2 depicts more detailed sequence of activities of the web plugin.

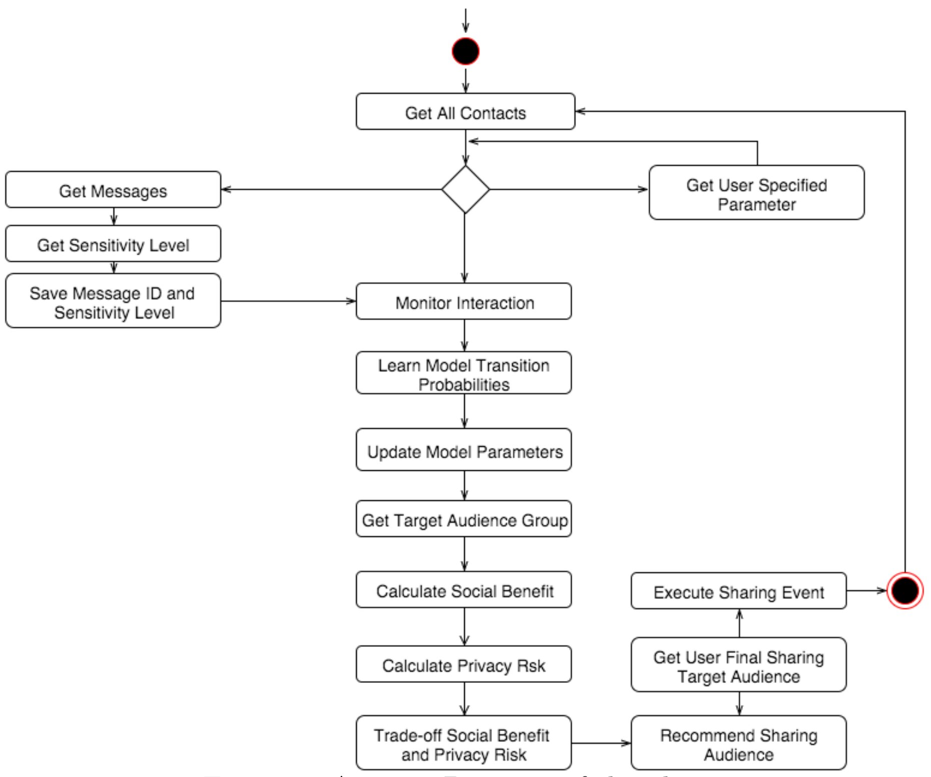

Get All Contacts
Get Messages
Get Sensitivity Level
Save Message ID and Sensitivity Level
Get User Specified Parameter
Monitor Interaction
Learn Model Transition Probabilities
Update Model Parameters
Get Target Audience Group
Calculate Social Benefit
Calculate Privacy Rsk
Trade-off Social Benefit and Privacy Risk
Recommend Sharing Audience
Get User Final Sharing Target Audience
Execute Sharing Event

Figure 3: Activity Diagram of the plugin

3.3 Monitor Design

Monitor is the key component of the back-end and is responsible for keeping track of interaction between the user and friends. Monitor class should fetch the user's friend list and follow-up all interaction a user may have during the plugin running time. Therefore, if somebody comments on a user's post or presses the Like button or re-shares it, the Monitor should be able to fetch the information related to this interaction including the ID of that person or the date and time of the interaction. Accordingly, a set of tools is required to communicate with Facebook database and access related data. These tools have generally been provided by Facebook.

3.3.1 Facebook Platform

The Facebook Platform is a set of tools and services provided by Facebook for software developers to create applications accessing data in Facebook database. Using this platform, third-party developers are provided with a set of programming interfaces that enable them to interact Facebook features.

3.3.2 Graph API

The Graph API is the backbone of Facebook Platform [Ref]. Technically, a social graph depicted personal relations within a social network. In this representation, nodes represent users and the friendship relationship is represented by edges.

Facebook Graph API is demonstrations of Facebook social graph so as objects, including people, pages, and events, and their connections, such as relationships and shared contents, are accessible by developers. As the structured view of data, the Graph API is the primary way to get data in and out of Facebook's platform.

Being a low-level HTTP-based API, working with the Facebook Graph API can be demanding. There are some libraries that can facilitate implementing this API.

3.3.3 Programming Languages and Libraries

The existing base code for the Online Learning Model is initially in Java. Moreover, the lack of system dependency of the language and widely availability made Java a reasonable choice for backend programming and led to less integration complexity. For employing Facebook Graph API,there are two prominent Java-based libraries:

- **RestFB:** An open source Facebook Graph API client written in Java. It provides some simple but powerful operative programming tools to fetch information from or publish new information to Facebook.
- **Facebook4J:** Another unofficial Java library for the Facebook Graph API. This library is also based on Java and provides some simple tools to integrate an application with the Facebook API.

For using these libraries, the related .jar file should only be added to the application class path. RestFB is more flexible and popular. There is also good documentation available for this library. Consequently, this library was chosen for this project.

3.3.4 Access Token

To have access to make graph API calls using the RestFB client, an access token is needed. An access token is a long string of characters that is an identification of the user and can be used by an app, in this case the plugin, to make graph API calls. Access tokens have different types and can be obtained via a number of methods. Within the scope of this project, UserAccess Token has been required. The user token is the most commonly used type of tokens that is required whenever an app calls an API to read or write a specific data on the user's Facebook. This token can be obtained via a login dialog and getting the user's permission so that enables the app to obtain the access token. It is also can be acquired manually through Graph API Explorer presented in the Facebook developers tools.

3.3.5 Implementation

The most important consideration in establishing the Monitor class is that it should not miss any interaction. It means that after running the plugin, every social interaction between the user and a friend should be taken into account in the learning procedure. Consequently, the Monitor should be efficient in performance because, in a possible scenario, the user may want to upload next information, immediately after the last one. In this case, the Monitor should be able to track the interactions on the most recent post, as well.
Furthermore, due to the research paper, in the learning to share approach, to evaluate each friend's behavior, the parameters should be calculated for each friend separately. Thus, in each learning procedure, the Monitor should be run n times, where n is the number of the user's friends. If this execution happens serially for a user who has many friends, it cannot be efficient enough to follow-up the recent interactions. Therefore, the Monitor should be developed in a multithreading manner in which a thread will be created per each friend and each of them is responsible to monitor interactions with that special friend.
There are two main possible ways to implement the Monitor. The first naïve way is going through all the information that the user has alreadyposted and calculating the parameters due to the interactions. Although this scheme is easy to implement, it is not feasible for users who havemany posts; if there are a lot of posts and interactions per each of them, the learning procedure will be computationally expensive and most of the recent interactions may be missed.
Another way is using Facebook notifications to keep track of the interactions. As it has already been explained, when an interaction happens, the user will get a notification. Thus, when the plugin starts to run, the Monitor should only listen to the notifications that the user gets. In this approach, at the beginning, it has been assumed that all the user's friends are privacy-aware without any malicious behavior. Then, as the time proceeds, the interactions will be captured by parsing the incoming notifications and the parameters will be calculated, consequently.

3.4 Structure of the plugin

The plugin consists of the following main elements:

- **Manifest:** a JSON-formatted file that includes the data and instructions about the plugin itself, such us the name, version and security settings.
- **Popup.js:** a JavaScript file responsible for the manual interaction with the plugin through the pop up.
- **Popup.html:** a file containing iframe to the web service.
- **Contentscript.js:** a file responsible for creation of an iframe for warning messages.

It must be mentioned that in order to provide with as much encapsulation of information as possible, the plugin component that ought to be installed by the user was developed to consist of the iframe tag solely, as opposed to containing any program logic. With such a choice, the iframe tag points to a relevant page, being the login, terms and conditions, sensitivity settings and a warning message that is served by the plugin web service. Such an approach not only ensured that the code is encapsulated, but it also made the overall design of the project much easier, as the communication between the monitor and the plugin was therefore possible without implementing additional communication services.

There were several other choices that were considered in this project as opposed to the usage of iframes. Here write it up if I manage to fix the issue

3.5 Structure of Web Service

The web service consists of the following elements:

- the backend server
- the frontend components
- database

Here flowchart

The backend server serves the frontend components in the order depicted by the Figure X. User information such as user id, set sensitivity level an message id (obtained from the monitor component) and user were then stored in the database to be reused accordingly.

3.6 Web Service Implementation

3.6.1 Backend

Web Service Host

The web service is hosted using AWS (Amazon Web Service) cloud computing platform . AWS delivers high availability performance, including bringing servers online and offline very quickly, and offering storage system that is capable of vast scaling. Moreover, AWS is very flexible, as all the services work and communicate together with your application to automatically judge demand and handle it accordingly . Other platforms considered for this project were Digital Ocean and Microsoft Azure, however AWS appealed as the most user-friendly, secure and flexible platform to all team members.

MVC Framework

As far as the Web Service is concerned, the MVC frameworks that were considered for this project included GWT (Google Web Toolkit), Spring and Spring Boot – a “simplified” MVC framework based on Spring. As this web service is the first web application attempted by all team members, it was important to select a framework that would be simple to learn and easy to use.

Although GTW seemed to be an easy to learn, well documented framework, according to research conducted by RebelLabs , out of 2164 JVM developers, over 70% stated that other frameworks must be used in addition to GTW to ensure the most optimal implementation of the service. That was not ideal for this project, as this would introduce additional complexity, risking the delivery of the final product within deadline.

Spring framework, on the other hand, was considered due to its popularity, reliability and flexibility related to settings available to support the project. However, such flexibility appeared to bring a huge disadvantage – it proved difficult and time consuming to set up the project as such. As a result, more research was conducted to find something as reliable and flexible as Spring, yet less difficult to configure and maintain. Spring Boot, therefore, became an obvious choice for this project, as it provides the shortest way to have a Spring web and it is a pre-configured, pre-sugared set of frameworks/technologies to reduce that boiler plate configuration.

Database
It was concluded after extensive research that MongoDB would be a perfect choice for this particular project. The main reason why such choice was made is the fact that it is easy to use and maintain and comes pre-set with Spring Boot. MongoDB is an open source NoSQL database that uses a document-oriented data model. Like other NoSQL databases, MongoDB uses dynamic schema, allowing the documents in a collection to have different fields and structures. The database stores data in JSON-like documents that can vary in structure. Related information is stored together for fast query access through the MongoDB query language.That made MongoDB a perfect choice for this particular project, as a majority of group members did not have any previous experience with SQL, therefore querying in Java was the best and fastest option.

3.7 Front End

3.7.1 The Design Process

The idea behind the overall design was to provide the user with sense of security by using calming colour pallet of blues with the mixture of purple. The design as such was also supposed to subtly remind the user of Facebook, ensuring at the same time that the plugin does not entirely look like a Facebook's product, which could have had potential legal implications.

Logo Development
The project required two logo designs – one to be used as an icon displayed in the browser's toolbar, and the second to be featured as a plugin's header. The icon's design had to be as clear as possible due to low visibility of the icon, whereas the header was required to provide the user with more information on the product.

The initial idea behind the icon was to create a shield-like shape to emphasise on the protection provided by the plugin, as well as containing "Learn To Share" initials and a red BETA flag. However, after using the icon on a tester plugin, it was observed that the LTS initials were completely unreadable. This discovery led to further research1 that inspired more minimalistic design that was not only visible, but also sent a clearer message of the plugin as a security tool for Facebook. The Beta flag and name of the product had to be discarded, as these were not visible at all in the Chrome toolbar.
As the header logo did not have visibility constraints, it was decided that the BETA flag and the product name should be displayed here instead. Thecolours were also changed to white, as the dark blue background provided with a contrast to the overall design of other web pages, drawing user's attention to the header in an instant.
Design of pages and warning message
The design of pages as such was aimed to remind the user of Facebook,therefore a similar white and grey colour pallet was applied. Moreover, as the plugin is a small-sized product, the best option was to apply a minimalistic design.
The design aim with regard to warning message, on the other hand, was to ensure it could stand out on user's Facebook page, without being too intrusive. Two designs were considered in particular: one featuring an exclamation mark in a shield, along with the message with Learn To Share header. It was concluded that the design with an exclamation mark would be too

Initially, the plugin was also meant to include the loading wheel to indicate to the user that the request is currently being processed. However, without AJAX one page web service, the loading wheel would cause the header to re-load frequently, providing with unpleasant user experience. Although discarded, the design can still be viewed in the Appedix Z.

3.8 Implementation and Design

Each page was designed to be implemented as a separate static web page. The structure of the front end was also considered to be developed as a one-page web service loaded using AJAX injections. That would allow the header to never require re-loading, allowing for better user experience. This alternative design would consist of a main page template, being a header and an empty container, as well as separate small page elements that would be loaded into the container accordingly. Although sounding very appealing, the idea was discarded as too risky to implement. The reasoning behind such was the fact that the plugin element was built on the basis of iframes, therefore introducing multiple iframe-like technologies could also introduce unnecessary bugs and could be hard to maintain and debug as such.

3.8.1 CSS Frameworks and Libraries

The purpose of the front-end CSS frameworks is to ensure the consistency across internal tools by providing with templates for HTML and CSS-based designs and optional JS extensions to animate certain web components. Before frameworks, various libraries were used for each component of the interface development separately, meaning inconsistencies and high maintenance1. Although the design required for the plugin was not extensive, it was concluded that in this case the implementation of a framework was the most optimal choice that would provide with the most professional look in the shortest amount of time. That would not be possible to achieve by first-time programmers by writing own CSS framework. After exploring several options, including Bootstrap, Semantic UI, Pure and Kube CSS, it was concluded that an open source Kube CSS would be the best choice for this project, as it provides with simple to use and beautiful to look at designs. Although not as popular as Bootstrap that has been favoured over 90,000 times on Github2, nor was developed by a well-known company, Kube CSS's minimalistic designs appear perfect for a small project like a plugin.

Additionally, jQuery's Slider3 along with Lugo Lab's Flat Slider4 libraries were used to achieve a perfect look for the Sensitivity Level slider.

3.8.2 JavaScript Frameworks

As the project included only a small section of JavaScript that was used for only for the sensitivity setting and AJAX requests, the use of a client-side JavaScript framework as such was not of a necessity. Although the CSS and HTML content for the plugin was also not extensive, the use of framework was most optimal choice in that instance, as CSS frameworks provide with.

Although frameworks like EmberJS or React are thought of as industry-standard practice, the setting time of the framework for such a small section would not be practical and therefore this part of the project was developed using vanilla JavaScript.

4 Methodology

4.1 Development Technique

Due to schedule constraints and lack of previous software development experience, the development process will take place iteratively[9]. Such approach allows for creation of mini-projects, which appears to be less risky than a spiral model in the instance where additional features would not be possible to implement.

Moreover, the project adopted the Agile ideology[10] with Scrum framework. The reasoning behind such is more flexible model of Scrum over XP, and a greater suitability for smaller, as in this instance, development teams. Also, Scrum will enable a more efficient development of the plugin by more structured process through division of work into tickets. To ensure the group maintains the agile development structure throughout the process, the use of Trello[11] software for the ticketing purposes, and Asana[12] for the project management was used.

A backlog was created on Trello in order to keep track of what needs to be done and a board for each phase to allow each member to update the status of their task as well as stay updated on each others progress.

4.2 Version Control

As far as the version control system is concerned, the distributed approach as opposed to centralized was adopted due to the flexibility it offers and the option to work on several features simultaneously[13]. Although it is believed that distributed model could be more difficult to maintain and prone to merge conflicts[13], previous experience with the Git-based software, along with daily meetings reduced such risks to minimum. Moreover, GitHub was decided upon over Mercurial and other distributed VCSs, due to reliability, popularity and an excellent logging capability[**?**].

4.3 Implementation

4.3.1 Monitor

The monitor was written in Java and utilises the RestFB client which provided methods to make calls to the Facebook Graph API to retrieve the data required. The methods were sufficient for the program to work. However, the access token for the corresponding user needs to be provided in order for the methods to work and this has to be retrieved by some other way.

Some users may have a large number of friends and posts, so retrieving data and processing it for all of these friends may be computationally expensive and slow. Concurrency was introduced into the program, so that one instance of the monitor runs for each friend concurrently. Furthermore, only the most recent posts were taken into account, so that the program does not have to look through so many posts.

For each post, the monitor identifies the interactions that the friend has made on the post as well as the level of sensitivity of the post and passes this information to the next component to perform the calculations for the learnt probabilities.

4.3.2 Adaptive Sharing Analyzer

The adaptive sharing analyzer takes in the learnt probabilities

4.4 Facebook Limitations

1. The user's notifications are required in the monitor to keep track of the interactions on the user's posts. However, the latest version (2.5) of the Facebook graph application programming interface (API)[15] does not allow access to the **manage_notifications** permission. Hence, a lower version (2.2) has to be used.

2. Only the Facebook notifications that have not been read by the user are obtainable through the API. Therefore, the program might miss some of the interactions if the user has already read them. A possible solution for this issue is employing Facebook webhooks which allows the plugin to become aware of new notifications.

3. Another problem with the notifications can happen when multiple instances of the same interaction occur on the same post by multiple friends; Facebook groups these interactions under the same notification ID which only provides the ID of the friend who interacted most recently. To solve this problem, the names of the friends who interacted were retrieved by parsing the notification messages.

4. Facebook recently introduced a new feature called reactions[16]. Other emotions can now be expressed for a post in addition to liking it. These reactions are ”Love”, ”Haha”, ”Wow”, ”Sad” and ”Angry”. Thus, the learning model has to be readjusted to fit these.

5. To create a plugin where the user can login to Facebook and give the plugin the required permissions, a proposal along with screenshots must be submitted to Facebook explaining how the permissions will be used, To do this, a prototype of the plugin that demonstrates the use of these permissions safely must first be created.

4.5 Plugin Component Issues

1. The current design of the plugin component as illustrated in Appendix A consists of two elements - the settings section and the warning message section[17]. The settings section element is only visible once user manually presses the browser's plugin button – using the “popup” approach. The second section is automatically triggered on mouse hover over a specific page element, therefore having knowledge of Facebook's DOM (Document Object Model). It was not known at the time that these two were considered as different types of extensions, and that both need to be synchronized with a complex messaging system[18]. The delay caused by the synchronisation could have been avoided if more thorough research took place initially.

2. The dynamic structure of Facebook's page poses a significant threat to reliability of the plugin development using the DOM-oriented approach, as it may prove impossible to target specific HTML tags for the triggering event due to ever-changing names of the elements. Additionally, it is possible that Facebook may not approve of manipulation of its DOM at all, which is currently being enquired about at Facebook. Therefore, there is a high possibility that splitting the plugin into two elements can be marked as a high-risk element or no longer feasible. An alternative solution may include moving the entire warning system to the sensitivity input section, providing with notification system that does not interfere with the Facebook's DOM.

3. After reading the Facebook Platform Policy[19], it was discovered that it is necessary to obtain user's consent prior to user's login to use Facebook API. Therefore the new requirement, being terms and conditions screen was added.

4.6 Testing Methodology

4.6.1 Unit testing

For the monitor component, the RestFB 1.19 API[20] for Java is used to make calls to Facebook's API to retrieve data. Therefore, a Black-Box approach was taken in order to test the methods provided by RestFB. Dummy Facebook accounts have been created and manual tests were written in JUnit[21] in order to check that each function defined by RestFB is able to provide the required data from the dummy accounts for the plugin. The testing process provided an understanding of the functionality of the API as well as its limitations, which was helpful in determining the best way to implement the methods.

For each of the other Java classes, a White-Box approach was taken. Synthetic data was used for each of the methods and the output was tested against the expected results. Each test case was written in JUnit and the synthetic data was generated to cover all branches in order to maximize code coverage. For example, the Adaptive Sharing Analyser takes in an array of learnt transition probabilities and the sensitivity level for a piece of information as inputs and returns one of three values (0.0 means do not share; 0.5 means give a warning message; 1.0 means safe to share). The transition probabilities were generated to simulate the different types of behaviours and each array of probabilities was tested with different levels of sensitivity and compared with the expected output. All the tests were put together into a test suite which checks the overall testing results. The JUnit test results can be found in Appendix B.

EclEmma[22], an Eclipse code coverage tool, was used to generate the code coverage report for each of the tests. The table below shows the code coverage of the tests.

Table 4: Code Coverage for Java classes

Class	Instruction Coverage	Branch Coverage
Monitor	98.9 %	100.0%
Transition Probability	92.5%	100.0%
Adaptive Sharing Analyser	94.3%	75.0%
Social Benefit Calculator	99.6%	83.3%
Privacy Risk Calculator	100.0%	100.0%
Overall Coverage	94.2%	87.0%

The vanilla JavaScript and JQuery unit tests will be completed using QUnit[23] once the integration of the plugin takes place, as the majority of the code communicates with Facebook and the server.

4.6.2 Accuracy Testing

As part of system testing, emphasis was placed on sensitivity and specificity[24]. The results from the tests helped to determine the best parameter values for the plugin to have an accurate determination of the respective friends' behaviours.

To do this, a test was be run for each parameter value used in the learning model. For each experiment, simulating either an immediate harmful behaviour pattern or delayed harmful behaviour pattern, and varying either the privacy risk threshold (W_i^{PR}) or the social benefit threshold (W_i^{SB}). Each experiment was run ten times and the average values were used.

Synthetically generated data was used to model different behaviours. For each behaviour, the test involved varying the parameter intended for testing and recording the number of true positive(TP), false positive(FP), true negative(TN) and false negative(FN) suggestions made by the system. Each kind of output is specified as follows:

- TP: the number of times the system guided the user to share the item of information with friend j, and j's actual re-share probability was within the acceptable range; the times the system guided the user not to share and both system requirements were violated;
- FP: the number of times the system flagged a warning and one of the two system requirements(W_i^{PR} or W_i^{SB}) had been violated;
- TN: the number of times the system guided the user not to share and only one of the system requirements was violated; the times the system guided the user to share the information with friend j, and j's actual re-share probability was above the acceptable range;
- FN: the number of times the system flagged a warning and both system requirements had been violated;

Based on the results from the above experiment, the following will be used to measure the system performance.
Accuracy(ACC) measures how accurately it determines the sharing behaviour

$$ACC : \frac{(\sum TP + \sum TN)}{(\sum FP + \sum FN + \sum TP + \sum TN)} \tag{1}$$

Sensitivity(TPR) measures the proportion of correctly identified positives

$$TPR : \frac{\sum TP}{(\sum TP + \sum FN)} \tag{2}$$

Specificity(SPC) meaning the proportion of correctly identified negatives formula

$$SPC : \frac{\sum TN}{(\sum FP + \sum TN)} \tag{3}$$

In each computing result, the higher the value, the more efficiently it learned.

5 Group Work

The project was divided into four parts: The Monitor, User Interface and Server Implementation, Obtaining Facebook Permissions and Testing Framework. Each member was assigned to one part as seen in the table below.

Table 5: Division of Work

Monitor	**User Interface and Server**	**Obtaining Facebook Persmissions**	**Testing Framework**
Boon Eng Oh Shahi Shahrokh	Magdalena Sadowska	Shanshan Fu	Yunxin Zhou Ani Petrosyan

For the monitor, Boon worked on integration of the monitor with the other parts of the plugin while Shahrokh worked on fetching the required data using the RestFB API.

6 Final Product

Unfortunately, due to Facebook's strict regulations, the permissions required to request the user for the required permissions through a user interface could not be obtained.

The final product is able to identify some behaviour patterns correctly and peform the appropriate action for these friends.

References

[1] eMarketer. 2014. *Internet to Hit 3 Billion Users in 2015.* [ONLINE]. Available at `http://www.emarketer.com/Article/Internet-Hit-3-Billion-Users-2015/1011602`[Accessed 23 January 16]

[2] Xu, H., Teo, H.H., Tan, B.C. and Agarwal, R., 2009. The role of push-pull technology in privacy calculus: the case of location-based services. *Journal of Management Information Systems*, 26(3), pp.135-174.

[3] Wang, Y., Leon, P.G., Chen, X. and Komanduri, S., 2013. From Facebook regrets to facebook privacy nudges. *Ohio St. LJ, 74*, p.1307.

[4] Wang, Y., Norcie, G., Komanduri, S., Acquisti, A., Leon, P.G. and Cranor, L.F., 2011, July. I regretted the minute I pressed share: A qualitative study of regrets on Facebook. *In Proceedings of the Seventh Symposium on Usable Privacy and Security* (p. 10). ACM.

[5] Statista. 2016. *Leading social networks worldwide as of January 2016, ranked by number of active users (in millions).*[ONLINE]. Available at `http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/`[Accessed 26 January 16].

[6] Yang, M., Yu, Y., Bandara, A.K. and Nuseibeh, B., 2014, September. Adaptive sharing for online social networks: a trade-off between privacy risk and social benefit. *In Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*(pp. 45-52). IEEE.

[7] Van Lamsweerde, A. and Letier, E., 2004. From object orientation to goal orientation: A paradigm shift for requirements engineering. In *Radical Innovations of Software and Systems Engineering in the Future* (pp. 325-340). Springer Berlin Heidelberg.

[8] Oracle. *The Java Language Environment.*[ONLINE] Available at: `http://www.oracle.com/technetwork/java/intro-141743.html#318` [Accessed 27 January 16].

[9] Chang, S.K., 2002. *Handbook of Software Engineering & Knowledge Engineering: Fundamentals.* (Vol. 1). World Scientific.

[10] Sutherland, J., Schwaber, K., Scrum, C.C.O. and Sutherl, C.J., 2007. *The scrum papers: Nuts, bolts, and origins of an agile process.*

[11] Trello.[ONLINE] Available at: `https://trello.com/`. [Accessed 01 February 16].

[12] Asana.[ONLINE] Available at: `https://app.asana.com/`. [Accessed 28 January 16].

[13] Hardeep, S., 2013, *Designing, Engineering, and Analyzing Reliable and Efficient Software.*

[14] Van Baarsen, J. 2014. *GitLab Cookbook.* Packt Publishing Ltd.

[15] Facebook for developers. 2010. Facebook Graph API. [ONLINE] Available at: `https://developers.facebook.com/docs/graph-api`. [Accessed 06 March 16].

[16] Facebook. 2006. Newsroom. [ONLINE] Available at: `http://newsroom.fb.com/news/2016/02/reactions-now-available-globally/`. [Accessed 06 March 16].

[17] Google Chrome. Choosing an App Type. [ONLINE] Available at: `https://developer.chrome.com/webstore/choosing`. [Accessed 05 March 16].

[18] Google Chrome. Message Passing. [ONLINE] Available at: `https://developer.chrome.com/extensions/messaging`. [Accessed 06 March 16].

[19] Facebook for developers. Platform Policy. [ONLINE] Available at: `https://developers.facebook.com/policy/`. [Accessed 01 March 16].

[20] RestFB 1.20.0. 2010. Overview. [ONLINE] Available at: `http://restfb.com/javadoc/`. [Accessed 05 March 16].

[21] Tahchiev, P., Leme, F., Massol, V. and Gregory, G., 2010. JUnit in action. Manning Publications Co..

[22] EclEmma. 2006. Java Code Coverage for Eclipse . [ONLINE] Available at: `http://eclemma.org/userdoc/`. [Accessed 03 March 16].

[23] Sheiko, D., 2013. Instant Testing with QUnit. Packt Publishing Ltd.

[24] Flach, P., 2012. Machine learning: the art and science of algorithms that make sense of data. Cambridge University Press.

A User Plugin Mockups

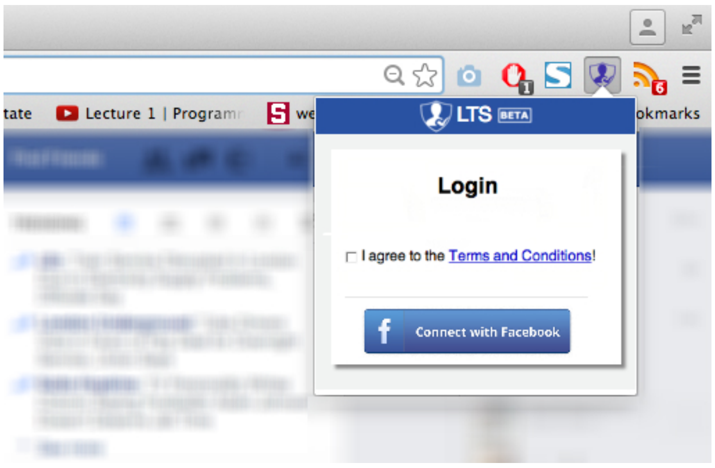


Figure 1: Login Screen

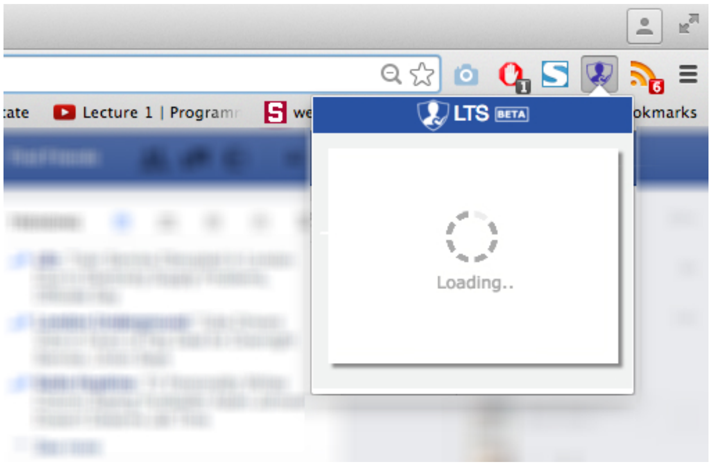


Figure 2: Loading Screen

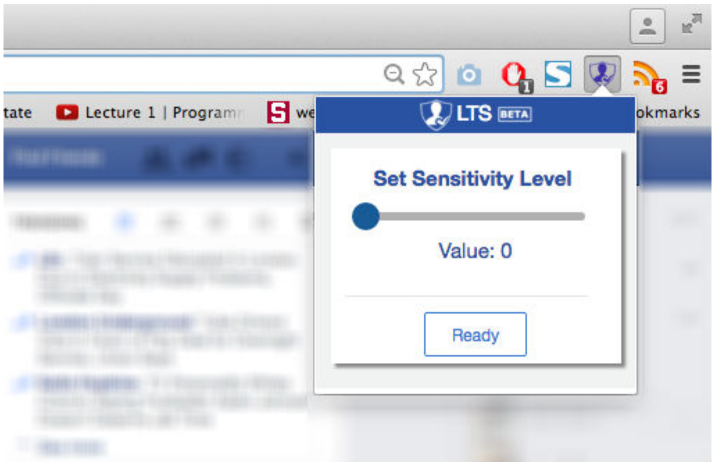


Figure 3: Sensitivity Input

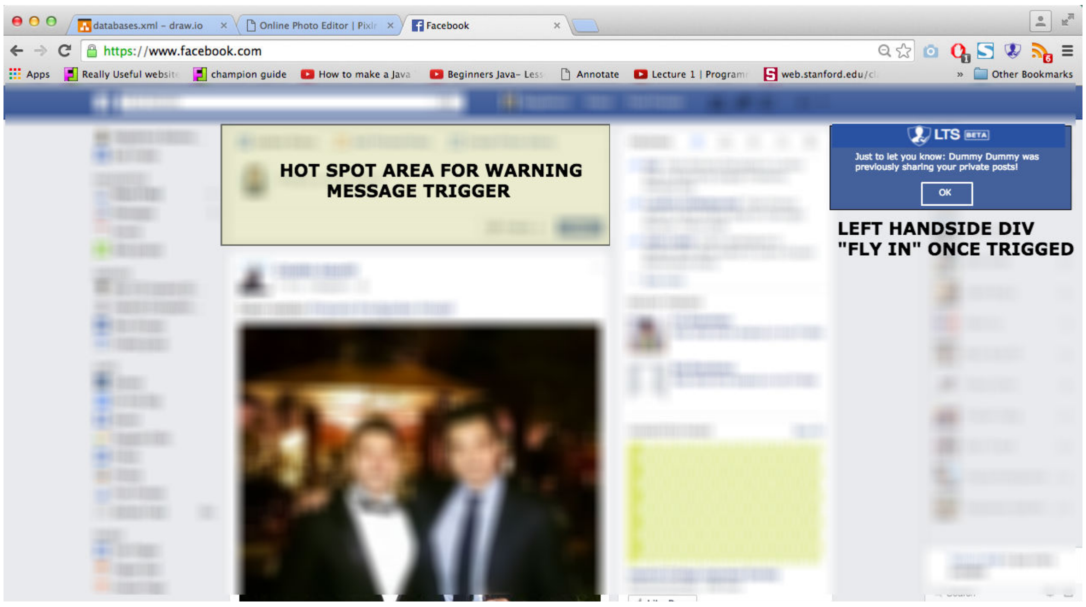
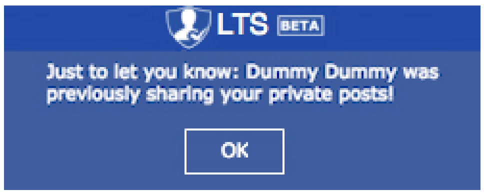
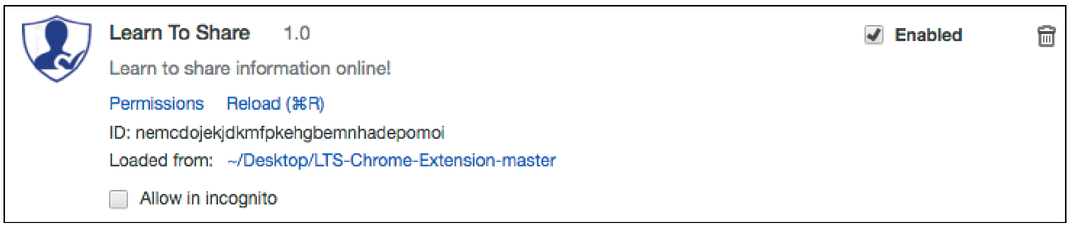

https://www.facebook.com
HOT SPOT AREA FOR WARNING MESSAGE TRIGGER
LTS BETA
Just to let you know: Dummy Dummy was previously sharing your private posts!
OK
LEFT HANDSIDE DIV "FLY IN" ONCE TRIGGED

Figure 4: Warning message



Just to let you know: Dummy Dummy was previously sharing your private posts!

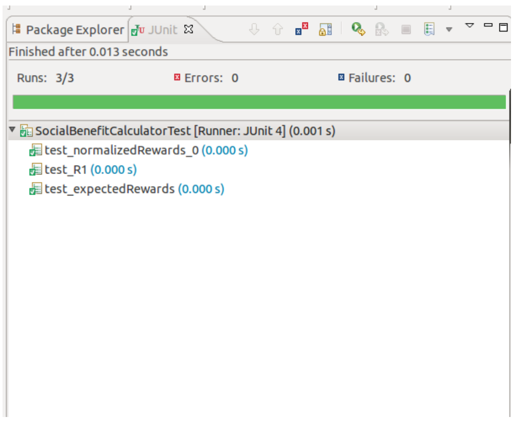
Figure 5: Warning Popup



Learn To Share 1.0
Enabled
Learn to share information online!
Permissions Reload (⌘R)
ID: nemcdojekjdkmfpkehgbemnhadepomoi
Loaded from: ~/Desktop/LTS-Chrome-Extension-master
Allow in incognito

Figure 6: Chrome Extension

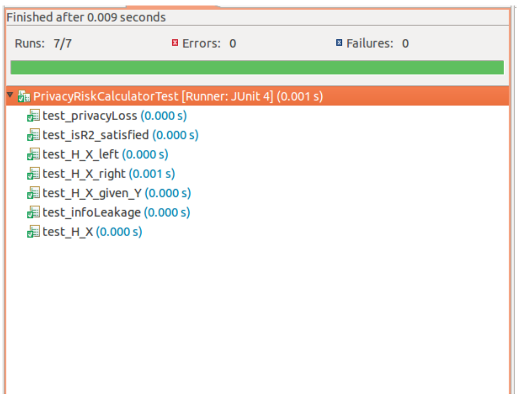
B Test Results



```
Package Explorer  JUnit
Finished after 0.013 seconds
Runs: 3/3        Errors: 0        Failures: 0

SocialBenefitCalculatorTest [Runner: JUnit 4] (0.001 s)
  test_normalizedRewards_0 (0.000 s)
  test_R1 (0.000 s)
  test_expectedRewards (0.000 s)
```

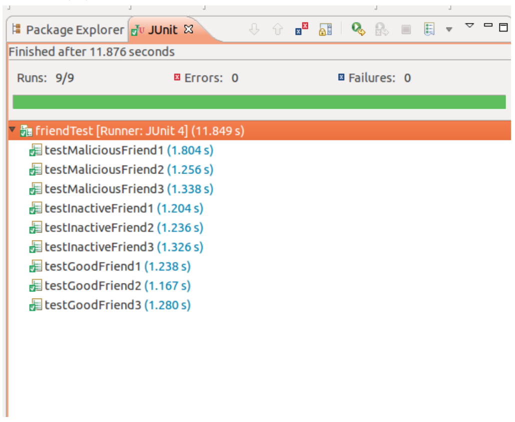
(a) Social Benefit Calculator Test



```
Finished after 0.009 seconds
Runs: 7/7        Errors: 0        Failures: 0

PrivacyRiskCalculatorTest [Runner: JUnit 4] (0.001 s)
  test_privacyLoss (0.000 s)
  test_isR2_satisfied (0.000 s)
  test_H_X_left (0.000 s)
  test_H_X_right (0.001 s)
  test_H_X_given_Y (0.000 s)
  test_infoLeakage (0.000 s)
  test_H_X (0.000 s)
```

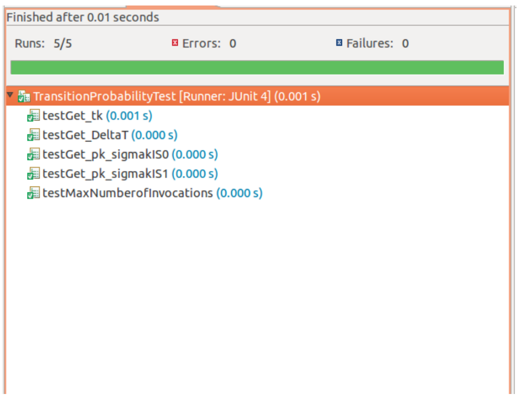
(b) Privacy Risk Calculator Test



```
Package Explorer  JUnit
Finished after 11.876 seconds
Runs: 9/9        Errors: 0        Failures: 0

friendTest [Runner: JUnit 4] (11.849 s)
  testMaliciousFriend1 (1.804 s)
  testMaliciousFriend2 (1.256 s)
  testMaliciousFriend3 (1.338 s)
  testInactiveFriend1 (1.204 s)
  testInactiveFriend2 (1.236 s)
  testInactiveFriend3 (1.326 s)
  testGoodFriend1 (1.238 s)
  testGoodFriend2 (1.167 s)
  testGoodFriend3 (1.280 s)
```

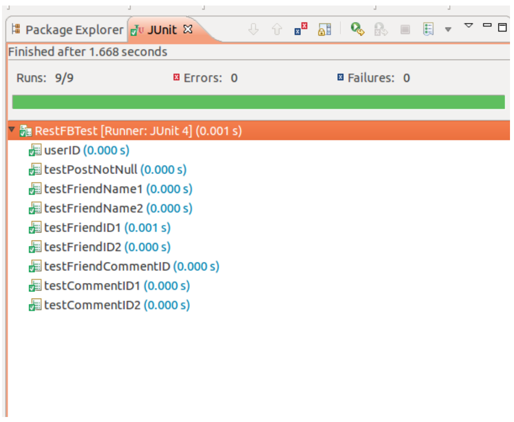
(c) Adaptive Sharing Analyser Test



```
Finished after 0.01 seconds
Runs: 5/5        Errors: 0        Failures: 0

TransitionProbabilityTest [Runner: JUnit 4] (0.001 s)
  testGet_tk (0.001 s)
  testGet_DeltaT (0.000 s)
  testGet_pk_sigmakIS0 (0.000 s)
  testGet_pk_sigmakIS1 (0.000 s)
  testMaxNumberofInvocations (0.000 s)
```

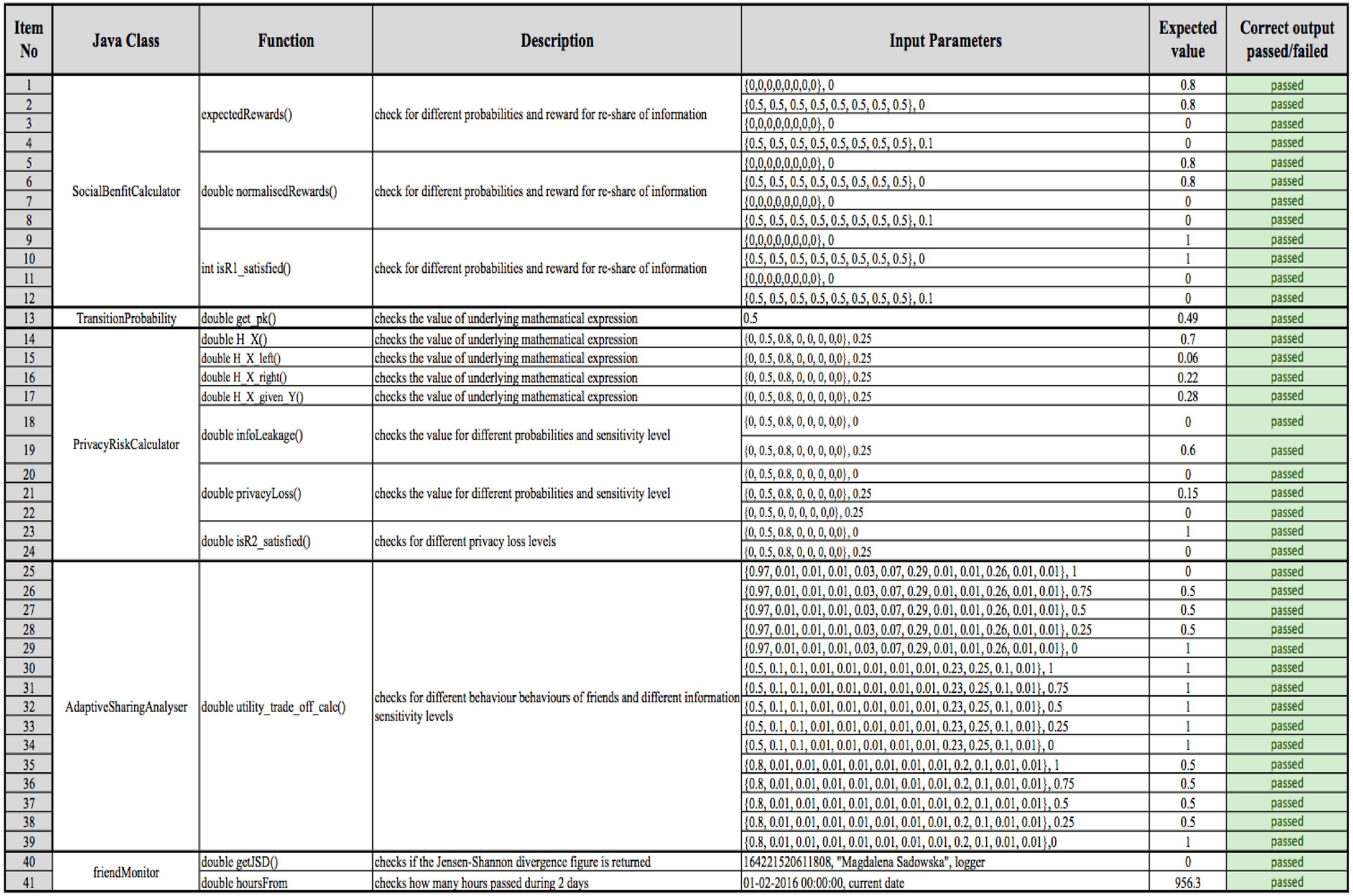
(d) Transition Probability Test



```
Package Explorer  JUnit
Finished after 1.668 seconds
Runs: 9/9        Errors: 0        Failures: 0

RestFBTest [Runner: JUnit 4] (0.001 s)
  userID (0.000 s)
  testPostNotNull (0.000 s)
  testFriendName1 (0.000 s)
  testFriendName2 (0.000 s)
  testFriendID1 (0.001 s)
  testFriendID2 (0.000 s)
  testFriendCommentID (0.000 s)
  testCommentID1 (0.000 s)
  testCommentID2 (0.000 s)
```

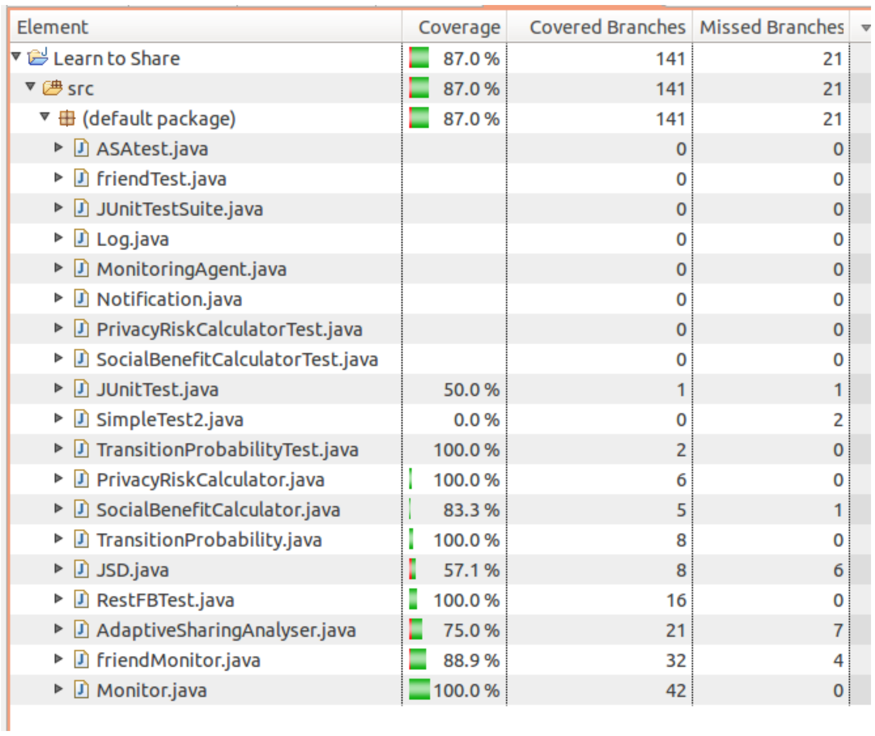
(e) RestFB Test



Item No	Java Class	Function	Description	Input Parameters	Expected value	Correct output passed/failed
1	SocialBenfitCalculator	expectedRewards()	check for different probabilities and reward for re-share of information	{0,0,0,0,0,0,0,0}, 0	0.8	passed
2				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0	0.8	passed
3				{0,0,0,0,0,0,0,0}, 0	0	passed
4				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0.1	0	passed
5		double normalisedRewards()	check for different probabilities and reward for re-share of information	{0,0,0,0,0,0,0,0}, 0	0.8	passed
6				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0	0.8	passed
7				{0,0,0,0,0,0,0,0}, 0	0	passed
8				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0.1	0	passed
9		int isR1_satisfied()	check for different probabilities and reward for re-share of information	{0,0,0,0,0,0,0,0}, 0	1	passed
10				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0	1	passed
11				{0,0,0,0,0,0,0,0}, 0	0	passed
12				{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}, 0.1	0	passed
13	TransitionProbability	double get_pk()	checks the value of underlying mathematical expression	0.5	0.49	passed
14	PrivacyRiskCalculator	double H_X()	checks the value of underlying mathematical expression	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.7	passed
15		double H_X_left()	checks the value of underlying mathematical expression	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.06	passed
16		double H_X_right()	checks the value of underlying mathematical expression	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.22	passed
17		double H_X_given_Y()	checks the value of underlying mathematical expression	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.28	passed
18		double infoLeakage()	checks the value for different probabilities and sensitivity level	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0	0	passed
19				{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.6	passed
20		double privacyLoss()	checks the value for different probabilities and sensitivity level	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0	0	passed
21				{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0.15	passed
22				{0, 0.5, 0, 0, 0, 0, 0,0}, 0.25	0	passed
23		double isR2_satisfied()	checks for different privacy loss levels	{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0	1	passed
24				{0, 0.5, 0.8, 0, 0, 0, 0,0}, 0.25	0	passed
25	AdaptiveSharingAnalyser	double utility_trade_off_calc()	checks for different behaviour behaviours of friends and different information sensitivity levels	{0.97, 0.01, 0.01, 0.01, 0.03, 0.07, 0.29, 0.01, 0.01, 0.26, 0.01, 0.01}, 1	0	passed
26				{0.97, 0.01, 0.01, 0.01, 0.03, 0.07, 0.29, 0.01, 0.01, 0.26, 0.01, 0.01}, 0.75	0.5	passed
27				{0.97, 0.01, 0.01, 0.01, 0.03, 0.07, 0.29, 0.01, 0.01, 0.26, 0.01, 0.01}, 0.5	0.5	passed
28				{0.97, 0.01, 0.01, 0.01, 0.03, 0.07, 0.29, 0.01, 0.01, 0.26, 0.01, 0.01}, 0.25	0.5	passed
29				{0.97, 0.01, 0.01, 0.01, 0.03, 0.07, 0.29, 0.01, 0.01, 0.26, 0.01, 0.01}, 0	1	passed
30				{0.5, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.23, 0.25, 0.1, 0.01}, 1	1	passed
31				{0.5, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.23, 0.25, 0.1, 0.01}, 0.75	1	passed
32				{0.5, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.23, 0.25, 0.1, 0.01}, 0.5	1	passed
33				{0.5, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.23, 0.25, 0.1, 0.01}, 0.25	1	passed
34				{0.5, 0.1, 0.1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.23, 0.25, 0.1, 0.01}, 0	1	passed
35				{0.8, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.2, 0.1, 0.01, 0.01}, 1	0.5	passed
36				{0.8, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.2, 0.1, 0.01, 0.01}, 0.75	0.5	passed
37				{0.8, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.2, 0.1, 0.01, 0.01}, 0.5	0.5	passed
38				{0.8, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.2, 0.1, 0.01, 0.01}, 0.25	0.5	passed
39				{0.8, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.2, 0.1, 0.01, 0.01},0	1	passed
40	friendMonitor	double getJSD()	checks if the Jensen-Shannon divergence figure is returned	16422152061 1808, "Magdalena Sadowska", logger	0	passed
41		double hoursFrom	checks how many hours passed during 2 days	01-02-2016 00:00:00, current date	956.3	passed

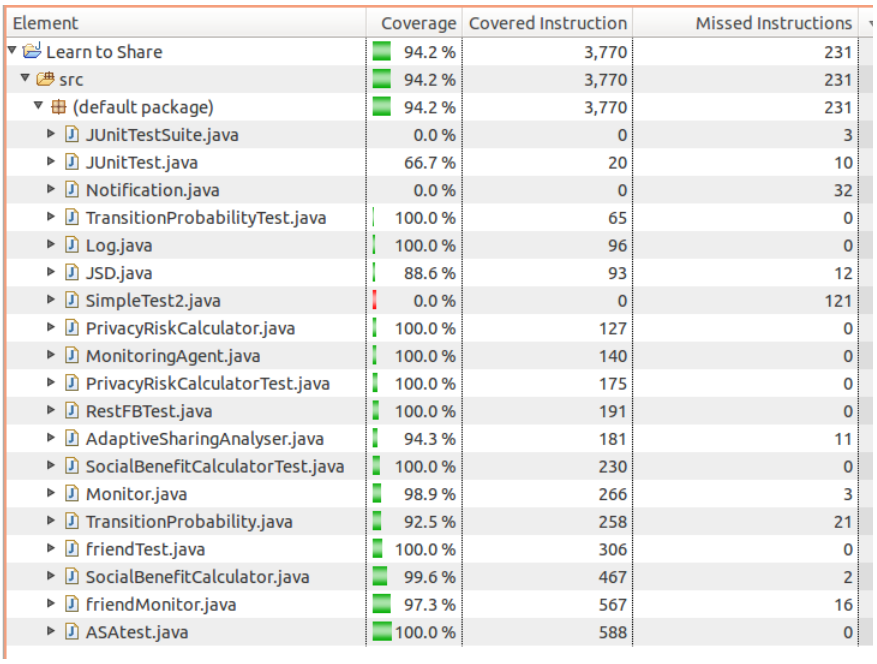
Figure 8: JUnit Results

C Code Coverage Report



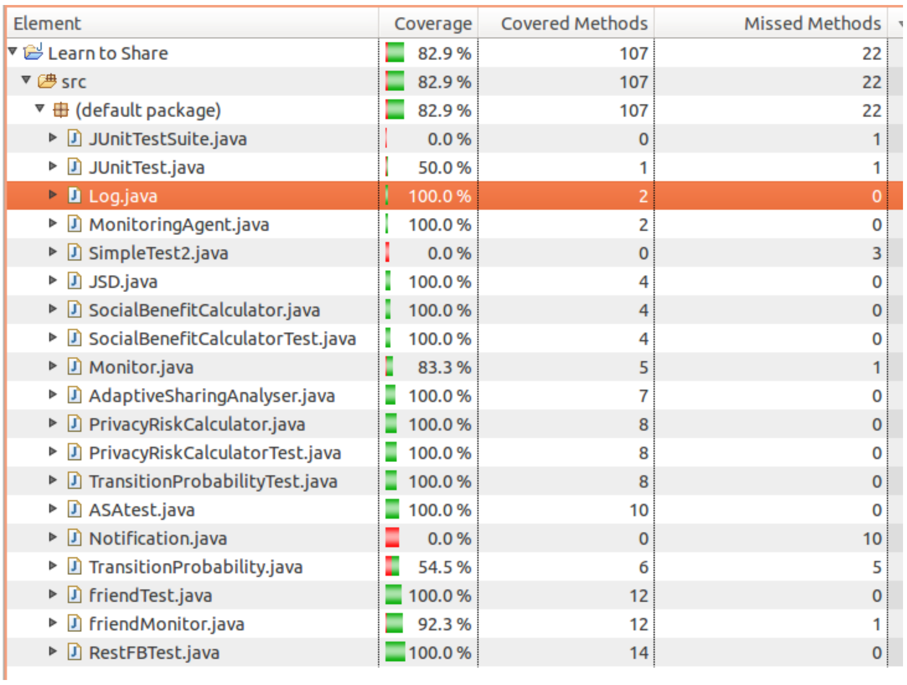
Element	Coverage	Covered Branches	Missed Branches
Learn to Share	87.0 %	141	21
src	87.0 %	141	21
(default package)	87.0 %	141	21
ASAtest.java		0	0
friendTest.java		0	0
JUnitTestSuite.java		0	0
Log.java		0	0
MonitoringAgent.java		0	0
Notification.java		0	0
PrivacyRiskCalculatorTest.java		0	0
SocialBenefitCalculatorTest.java		0	0
JUnitTest.java	50.0 %	1	1
SimpleTest2.java	0.0 %	0	2
TransitionProbabilityTest.java	100.0 %	2	0
PrivacyRiskCalculator.java	100.0 %	6	0
SocialBenefitCalculator.java	83.3 %	5	1
TransitionProbability.java	100.0 %	8	0
JSD.java	57.1 %	8	6
RestFBTest.java	100.0 %	16	0
AdaptiveSharingAnalyser.java	75.0 %	21	7
friendMonitor.java	88.9 %	32	4
Monitor.java	100.0 %	42	0

Figure 1: Branch Coverage



Element	Coverage	Covered Instruction	Missed Instructions
Learn to Share	94.2 %	3,770	231
src	94.2 %	3,770	231
(default package)	94.2 %	3,770	231
JUnitTestSuite.java	0.0 %	0	3
JUnitTest.java	66.7 %	20	10
Notification.java	0.0 %	0	32
TransitionProbabilityTest.java	100.0 %	65	0
Log.java	100.0 %	96	0
JSD.java	88.6 %	93	12
SimpleTest2.java	0.0 %	0	121
PrivacyRiskCalculator.java	100.0 %	127	0
MonitoringAgent.java	100.0 %	140	0
PrivacyRiskCalculatorTest.java	100.0 %	175	0
RestFBTest.java	100.0 %	191	0
AdaptiveSharingAnalyser.java	94.3 %	181	11
SocialBenefitCalculatorTest.java	100.0 %	230	0
Monitor.java	98.9 %	266	3
TransitionProbability.java	92.5 %	258	21
friendTest.java	100.0 %	306	0
SocialBenefitCalculator.java	99.6 %	467	2
friendMonitor.java	97.3 %	567	16
ASAtest.java	100.0 %	588	0

Figure 2: Instruction Coverage



Element	Coverage	Covered Methods	Missed Methods
Learn to Share	82.9 %	107	22
src	82.9 %	107	22
(default package)	82.9 %	107	22
JUnitTestSuite.java	0.0 %	0	1
JUnitTest.java	50.0 %	1	1
Log.java	100.0 %	2	0
MonitoringAgent.java	100.0 %	2	0
SimpleTest2.java	0.0 %	0	3
JSD.java	100.0 %	4	0
SocialBenefitCalculator.java	100.0 %	4	0
SocialBenefitCalculatorTest.java	100.0 %	4	0
Monitor.java	83.3 %	5	1
AdaptiveSharingAnalyser.java	100.0 %	7	0
PrivacyRiskCalculator.java	100.0 %	8	0
PrivacyRiskCalculatorTest.java	100.0 %	8	0
TransitionProbabilityTest.java	100.0 %	8	0
ASAtest.java	100.0 %	10	0
Notification.java	0.0 %	0	10
TransitionProbability.java	54.5 %	6	5
friendTest.java	100.0 %	12	0
friendMonitor.java	92.3 %	12	1
RestFBTest.java	100.0 %	14	0

Figure 3: Method Coverage

D Logbook