PROJECT ICEWOLF

A WEB BROWSER POWERED BY JAVA

Fall Semester 2017

N. Ivanov, G. Bowen, D. Parsons

Southwest Minnesota State University

COMP 425 – Software Engineering

Department of Computer Science

Southwest Minnesota State University
Fall 2017 Semester

Class: COMP 425-01 "Software Engineering"
Professor: Kourosh Mortezapour

Team Project: Icewolf web-browser
Team Members: Nikolay Ivanov, Gregory Bowen, Dylan Parsons

OPPORTUNITY STUDY

Date: September 13, 2017

Table of Contents

Problem Statement

According to StatCounter Global Stats, the most popular web browsers are Google Chrome, Apple Safari, UC Browser, Mozilla Firefox, Opera, Microsoft InternetExplorer, Samsung Internet, Google Android/Android Chrome, and Microsoft Edge[1].

These web browsers have the following features in common:

- Most of them are either free open source or freeware.
- All of them are stuffed with sophisticated features.
- Most of them support plugins or activation of additional features.

However, all these browsers are lacking the following:

- None of them is built using JavaFX, the successor of AWT and Swing. Distributed by Oracle along with Java SE 8[2], JavaFX is intended to be the main GUI toolkit for cross-platform desktop applications written in Java and Kotlin.
- None of the aforementioned browsers have an easily adaptable source codebase. In other words, companies and private users can't easily re-brand, adjust, and re-distribute their own forks of the web browsers listed above.
- Neither of the aforementioned browsers, due to high level or abstraction and the complexity of their codebases, is capable of being an easy-to-use learning or fundamental platform for software developers and Computer Science students.

Therefore, it is important to design and develop a cross-platform lightweight web-browser based on JavaFX framework. The browser should have developer-ready well-organized source code and should be licensed under a permissive open-source license. The web-browser should have a modular organization that would allow developers to easily implement new features for the browser without utilizing a complicated (and sometimes unsafe) plugin infrastructure.

Functions to be Provided

The following functions will be provided by the browser:

- GUI Interface
- Cross-platform: Linux/Windows/Mac
- Full HTML5 Support
- Full CSS3 Support
- Full JavaScript/ECMA Support
- Modular structure
- Menu bar
- Universal address bar
- HTTP/HTTPS-protocol auto-recognition

---

[1] http://gs.statcounter.com/browser-market-share#monthly-201708-201708-bar

[2] The only exception is the distribution of Java SE for ARM architecture.

Processing Environment

The processing environment for the app covers all GUI desktop systems supporting Java SE 8 and JavaFX, including but not limited to, GNU/Linux, Microsoft Windows, and Apple Mac OS X/Sierra. Hardware.

User Characteristics

The user audience for Icewolf web browser is two-faceted. On the one hand, these are software developers who will use the open-source code of the web browser for educational or corporate purposes (building new features, forks, creating a corporate "brand" version of the browser). The browser is distributed under Apache 2.0 License, which is open-source and permissive. This allows everybody to adjust the browser to their needs without legal obstacles. On the other hand, the end users of the browser (clients, employees, students, etc.) is the other part of the audience.

The users should be running the browser in the desktop environment of either Linux, Mac OS X/Sierra or Microsoft Windows operating system. The recommended configuration of the desktop system running the browser should include at least 1GHz CPU, 2GB RAM and at least 2 GB of free disk space.

Solution Strategy

In order to implement the project, our team is going to use the Agile and Extreme Programming methodologies with Scrum meetings, 2-week iterations, comprehensive multi-faceted unit testing, and inter-team role distribution.

The following is the Agile Development Calendar for the project:

STAGE	DATES	ACTIVITIES	EST. HOURS OF WORK
Pre-development	8/30 – 9/11	• Ideas Brainstorming • Introductory Meeting • Pre-planning Meeting • Opportunity Study	10 h.
Iteration 1: Core Infrastructure	9/11 – 9/25	• Iteration 1 Planning Meeting [1] • Iteration 1 Scrum Meeting 1 [1/8] • Iteration 1 Scrum Meeting 2 [1/8] • Iteration 1 Scrum Meeting 3 [1/8] • Iteration 1 Scrum Meeting 4 [1/8] • Iteration 1 Scrum Meeting 5 [1/8] • Iteration 1 Scrum Meeting 6 [1/8] • Iteration 1 Scrum Meeting 7 [1/8] • Iteration 1 Scrum Meeting 8 [1/8] • Iteration 1 Individual Development [4]	13 h.

		• Iteration 1 Pair Development [3] • Iteration 1 Testing [1] • Iteration 1 Documentation [1] • Iteration 1 Paperwork and Storyboard Maintenance [1] • Iteration 1 Release [1]	
Iteration 2: First Module Implementation	9/25 – 10/9	• Iteration 2 Planning Meeting • Iteration 2 Scrum Meeting 1 • Iteration 2 Scrum Meeting 2 • Iteration 2 Scrum Meeting 3 • Iteration 2 Scrum Meeting 4 • Iteration 2 Scrum Meeting 5 • Iteration 2 Scrum Meeting 6 • Iteration 2 Scrum Meeting 7 • Iteration 2 Scrum Meeting 8 • Iteration 2 Bug Fixes [1] • Iteration 2 Individual Development [3] • Iteration 2 Pair Development • Iteration 2 Testing • Iteration 2 Documentation • Iteration 2 Paperwork and Storyboard Maintenance • Iteration 2 Release	13 h.
Iteration 3: Second Module Implementation	10/9 – 10/23	• Iteration 3 Planning Meeting • Iteration 3 Scrum Meeting 1 • Iteration 3 Scrum Meeting 2 • Iteration 3 Scrum Meeting 3 • Iteration 3 Scrum Meeting 4 • Iteration 3 Scrum Meeting 5 • Iteration 3 Scrum Meeting 6 • Iteration 3 Scrum Meeting 7 • Iteration 3 Scrum Meeting 8 • Iteration 3 Bug Fixes • Iteration 3 Individual Development • Iteration 3 Pair Development • Iteration 3 Testing • Iteration 3 Documentation • Iteration 3 Paperwork and Storyboard Maintenance • Iteration 3 Release	13 h.

Iteration 4: Third Module Implementation	10/23 – 11/6	• Iteration 4 Planning Meeting • Iteration 4 Scrum Meeting 1 • Iteration 4 Scrum Meeting 2 • Iteration 4 Scrum Meeting 3 • Iteration 4 Scrum Meeting 4 • Iteration 4 Scrum Meeting 5 • Iteration 4 Scrum Meeting 6 • Iteration 4 Scrum Meeting 7 • Iteration 4 Scrum Meeting 8 • Iteration 4 Bug Fixes • Iteration 4 Individual Development • Iteration 4 Pair Development • Iteration 4 Testing • Iteration 4 Documentation • Iteration 4 Paperwork and Storyboard Maintenance • Iteration 4 Release	13 h.
Iteration 5: Fourth Module Implementation	11/6 – 11/20	• Iteration 5 Planning Meeting • Iteration 5 Scrum Meeting 1 • Iteration 5 Scrum Meeting 2 • Iteration 5 Scrum Meeting 3 • Iteration 5 Scrum Meeting 4 • Iteration 5 Scrum Meeting 5 • Iteration 5 Scrum Meeting 6 • Iteration 5 Scrum Meeting 7 • Iteration 5 Scrum Meeting 8 • Iteration 5 Bug Fixes • Iteration 5 Individual Development • Iteration 5 Pair Development • Iteration 5 Testing • Iteration 5 Documentation • Iteration 5 Paperwork and Storyboard Maintenance • Iteration 5 Release	13 h.
Post-development	11/20 – 12/6		15 h.

Product Features

- Bright theme/Dark theme switcher;
- JavaScript enable/disable switcher.

Acceptance Criteria

When the project is finished, the following functionality should be tested and accepted by the instructor and some classmates:

- Interface usability;
- The ability to load page and navigation from one page to another;
- The ability to support modern standards: HTML5, CSS3 and JavaScript.
- The ability to work in different operating systems (Linux, Mac, Windows).

Sources of Information

Class Textbook:

Schach, Stephen R. *Object-oriented and classical software engineering*. New York: McGraw-Hill, 2011.

Self-purchased Kindle Book:

Langr, Jeff, Andrew Hunt, and David Thomas. *Pragmatic unit testing in Java 8 with JUnit*. Dallas, TX: The Pragmatic Bookshelf, 2015.

E-book available at SMSU McFarland Library:

NetLibrary, Inc., Ioannis G. Stamelos, and Panagiotis Sfetsos. *Agile Software Development Quality Assurance*. Hershey, PA: Information Science Reference, 2007.

Websites:

- https://en.wikipedia.org/wiki/Agile_software_development
- https://en.wikipedia.org/wiki/Scrum_(software_development)
- https://en.wikipedia.org/wiki/Extreme_programming
- https://en.wikipedia.org/wiki/Unit_testing

Glossary of Terms

Agile software development – describes a set of values and principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. These principles support the definition and continuing evolution of many software development methods[3].

Scrum is a framework for managing software development. Scrum is designed for teams of three to nine developers who break their work into one-week to maximum four-week cycles, called "sprints", check progress daily in 15-minute stand-up meetings, and deliver workable software at

[3] https://en.wikipedia.org/wiki/Agile_software_development

the end of every sprint. Approaches to coordinating the work of multiple scrum teams in larger organizations include Large-Scale Scrum and Scrum of Scrums[4].

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[5].

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use[6].

---

[4] https://en.wikipedia.org/wiki/Scrum_(software_development)

[5] https://en.wikipedia.org/wiki/Extreme_programming

[6] https://en.wikipedia.org/wiki/Unit_testing

Southwest Minnesota State University
Fall 2017 Semester

Class: COMP 425-01 “Software Engineering”
Professor: Kourosh Mortezapour

Team Project: Icewolf web browser
Team Members: Nikolay Ivanov, Gregory Bowen, Dylan Parsons

PROJECT SPECIFICATIONS

Updated: October 2, 2017

Table of Contents

Chapter 1: Introduction

Icewolf is a cross-platform lightweight web-browser based on JavaFX framework. The browser has a developer-ready well-organized source code and is licensed under a permissive open-source license[1]. The web-browser has a modular organization that would allow developers to easily implement new features for the browser without utilizing a complicated (and sometimes unsafe) plugin infrastructure.

Chapter 2: Environment Characteristics

2.1. Hardware

Icewolf is intended to be run on Desktop systems as well as mobile system with desktop platform support, such as Microsoft Surface, with the following minimal hardware requirements:

Minimal Configuration[2]

CPU: 1 GHz
RAM: 1 GB
Storage (HDD or SSD): 200 MB of free storage.
Screen Resolution: 1280x1024

Recommended Configuration

CPU: 1.4 GHz
RAM: 2 GB
Storage (HDD or SSD): 1 GB of free storage.
Screen Resolution: 1600x1200

Icewolf web browser is not intended to be run on mobile platforms, except platform intended by design to support desktop applications, such as Microsoft Surface. Although Raspberry Pi 2, Raspberry Pi 3, and Raspberry Pi, depending on their sub-versions, might show inferior configuration than the minimal requirements for the desktop environment, the internal optimizations of the operating systems' distributions designed for Raspberry Pi, allow to overcome the hardware limitations. With that being said, the three versions of Raspberry Pi listed above are capable of running Icewolf web browser, if JavaFX is available on the host operating system.

---

[1] https://www.apache.org/licenses/LICENSE-2.0

[2] Both the Minimal and the Recommended configurations has been tested using Oracle Virtual Box hardware emulator running under Mac OS X High Sierra as a host operating system and using Ubuntu 16.4 LTS and Microsoft Windows 10 as guest operating systems.

2.2. Peripherals

Minimal Configuration:

- QWERTY Keyboard
- Mouse, touchpad or trackball

Recommended Configuration:

- QWERTY Keyboard
- Mouse, touchpad or trackball
- Sound card and speakers

2.3. People

The browser is intended for a broad audience of users, from corporate graphic terminal low-skilled users to professional software developers building their own modules using Icewolf as a development platform. Causal users can use the browser as well as a simple browser (for example, when they don't have enough resources to run a fully-fledged web browser like Mozilla Firefox or Google Chrome). Although the primary intent for creating Icewolf web-browser is not to accommodate the causal users, they still could benefit from using the browser.

Chapter 3: Fixed Interfaces

3.1. Devices

No specific device requirements exist for Icewolf web-browser.

3.2. Operating System

The browser is cross-platform. It can be run in the following operating systems:

- Any desktop distribution of GNU/Linux with kernel version 3.8 and higher[3];
- Mac OS X 10.2+ or any version of Mac OS Sierra;
- Microsoft Windows 7, 8, or 10.

---

[3] Since JavaFX is not supported by Oracle Corporation on ARM architecture, the support of Icewolf browser on Raspberri Pi/Raspbian is limited.

3.3. Database

SQLite3 File-based database is used for storing bookmarks and settings. No extra setup or server configuration needed for using the database. The database allows to efficiently and structurally store bookmarks and settings when the app is not running.

The bookmarks are stored in two tables (see *Figure 1*) connected by a foreign key relationship. The settings are stored in a single table, and only first row of this database is used (see *Figure 1*).

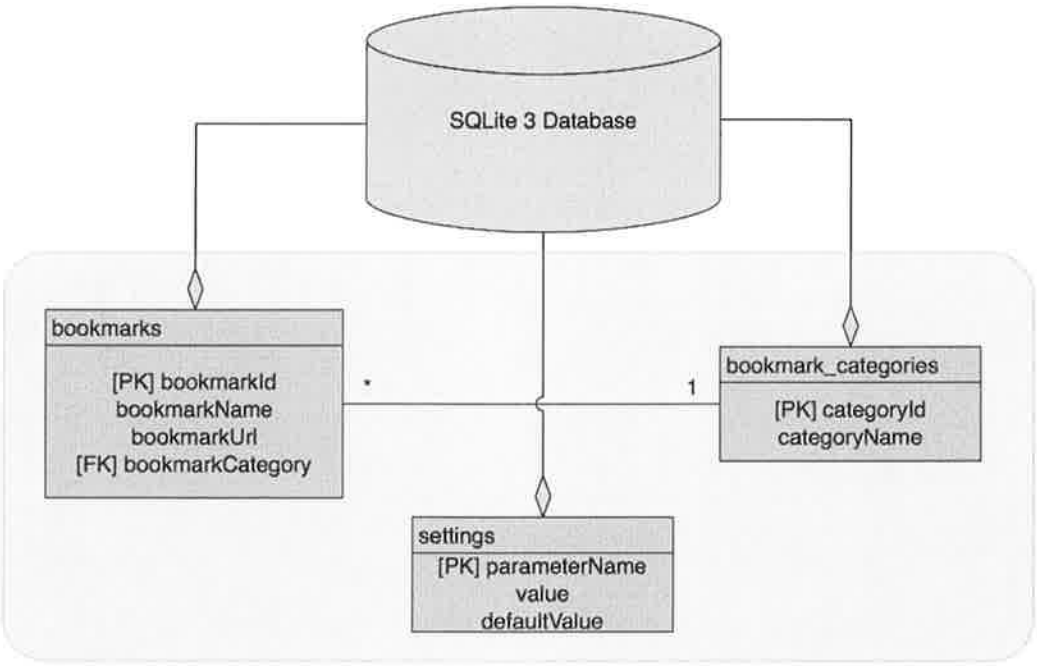


Figure 1. Icewolf SQLite 3 Database Design

Chapter 4: Input and Output

4.1. User Displays and Report Format

The browser uses JavaFX GUI interface, so all input, output and error handling is done according to Java Look and Feel Design Guidelines[4].

[4] http://www.oracle.com/technetwork/java/index-136139.html

4.2. User Command Summary

All commands from the user are entered through the GUI JavaFX interface. The following widgets are used in the app to receive input from the user (see *Figure 2*):

- Input text fields (address field, search field)
- Buttons
- Checkboxes
- Radio-buttons
- Drop-down lists
- Tabs
- Main menu

This list doesn't include possible widgets and other interfaces that are part of the dynamic web-pages the user might load in the web browser. The reason of such an exclusion is simple: any web page loaded in the browser is not a part of the browser.

The browser doesn't support any keyboard shortcuts in the "bare bone" version. However, the modular developer-ready design of the browser allows to implement keyboards shortcuts easily using JavaFX framework.

In the "bare bone" version of the browser there is no JavaScript console, but it is easy to implement one, if it is needed. Therefore, Icewolf doesn't have any command-line interface interaction with the user. All the available interactions are comprised to the set of input/output widgets listed in the *Figure 2*.

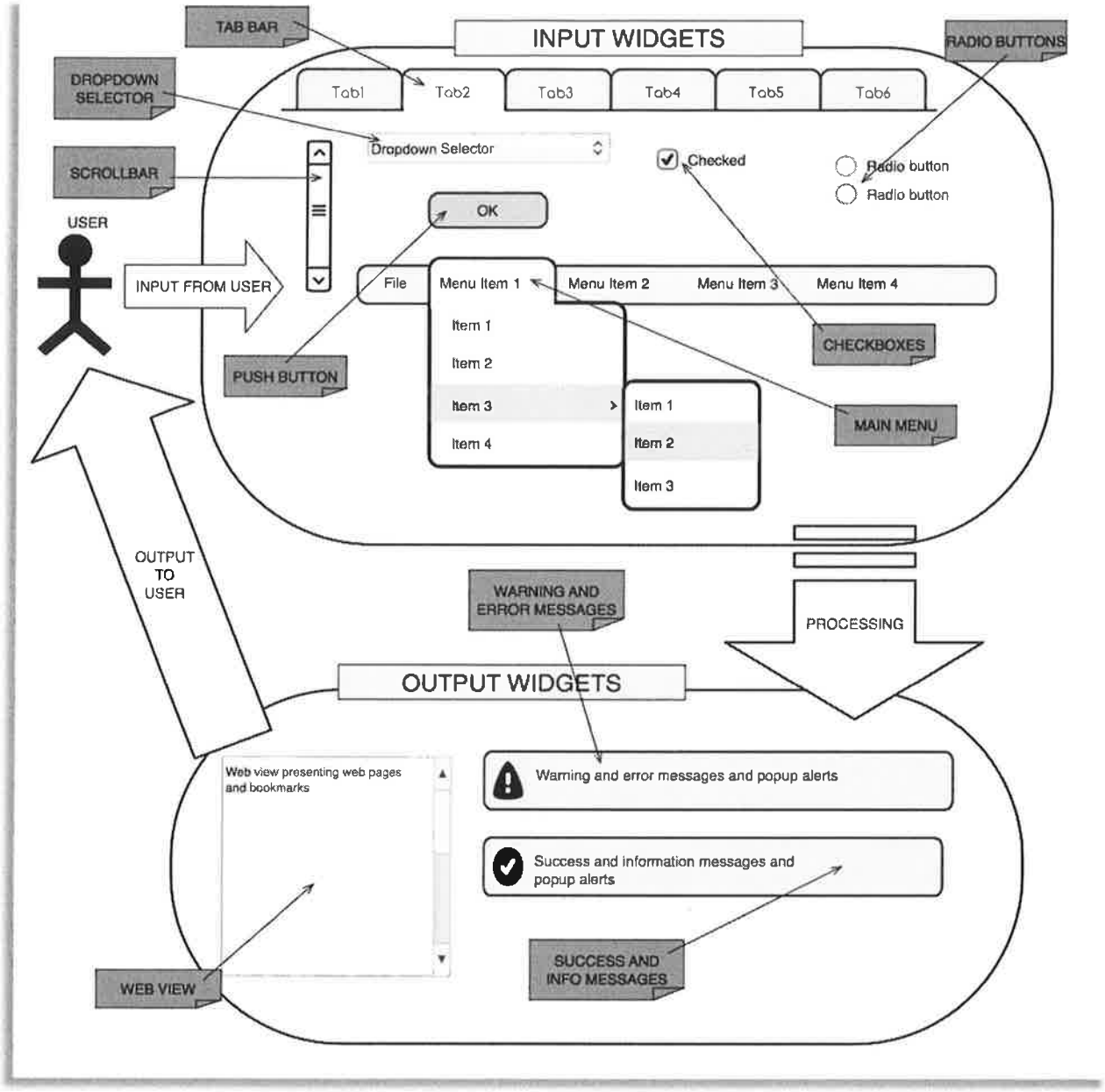


Figure 2. Icewolf User Interaction Workflow

Chapter 5: Modes

The following are the modes that the software can operate in:

- Startup mode;
- Web browsing/web search mode;
- Bookmark browsing mode;
- Configuration mode.

The modes are described in detail as follows.

5.1. Startup Mode

Startup mode is the mode when Icewolf browser has just been loaded and is waiting for the user to load a page, use a search bar, enter the bookmark manager, or launch the configuration dialog.

5.2. Web Browsing/Web Search Mode

The browser switches to this mode when the user either requests a web address or enters a search request in the search bar. The mode begins with loading the requested page or a default search engine with the given search request. When the page is loaded, the browser remains in this mode until the user decides to browse bookmarks, launch the configuration dialog or quit the application.

5.3. Bookmark Browsing Mode

The bookmark browsing mode is launched when the user selects Menu --> Bookmarks menu bar command. This command opens the list of bookmarks in the Web View area of the browser. Clicking one of the bookmarks automatically switches the browser from the Bookmark Browsing Mode into Web Browsing/Web Search Mode.

5.4. Configuration Mode

The configuration mode is a mode in which the main browser window is inactive, and the configuration dialog is running in the foreground. Browsing web pages or bookmarks is not possible in this mode until the settings are saved or cancelled and the dialog window is closed.

Chapter 6: Software Functions

The software has the following functions:

- Loading web pages;
- Supporting CSS3/JavaScripts/HTML5;
- Saving and using bookmarks;
- Searching using the search bar;
- Altering the program's default behavior by configuring it.

Chapter 7: Constraints and Goals

Icewolf web browser has the following constraints:

- In order to prevent the main menu from hiding, the minimum browser's window resolution is set to 800x600.

Icewolf web browser has the following goals:

- All pages weighing up to 128 MB should load within 12 seconds with the real bandwidth between the page and the server at least 32 Mbps.
- All functions should work properly with different screen resolutions and with different supported operating systems.

Chapter 8: Quantitative Requirements

8.1. Timing

- The page loading timeout, as well as the search result loading timeout is set to 2 minutes (120 seconds). If the time is out and the page (or search) result has not yet loaded, an error message will appear saying that the page cannot be loaded.

8.2. Precision

No precision requirements are present from Icewolf web-browser.

8.3. Other

- An attempt to load a page heavier than 128 MB will be interrupted to prevent overuse of the system resources;
- An attempt to create more than 1024 bookmarks will be interrupted in order to prevent overloading of the bookmark SQLite3 database;
- An attempt to type a URL/Address/Search request longer than 256 bytes will be interrupted to prevent potential code injection attack.

Chapter 9: Response to Undesired Events

The following is a list of undesired events that are handled by the browser and a pre-programmed response to these events:

- Out of memory -- an error message in a pop-up error window.
- Page can't be loaded -- an error message in the main WebView.
- The address bar input doesn't mean minimum requirements that would allow to guess the desired URL without ambiguity -- an error message in a popup window.

Chapter 10: Life Cycle Considerations

The following is the chart of Icewolf's lifecycle:

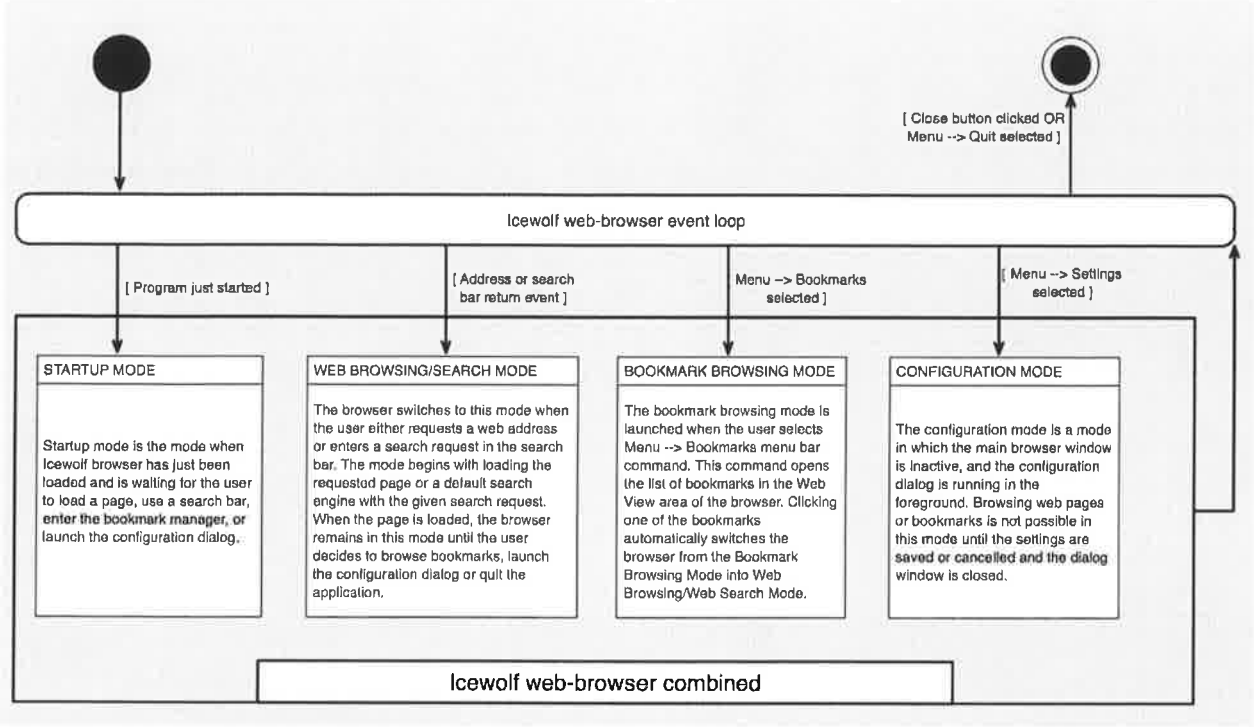


Figure 3. Statechart for Icewolf Web Browser

Chapter 11: Glossary

JavaFX: JavaFX is a software platform for creating and delivering desktop applications that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS[5].

Raspberry Pi: The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

SQLite: SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

---

[5] More information here:
https://en.wikipedia.org/wiki/JavaFX
http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html

Web Browser: A software program that allows the user to find and read encoded documents in a form suitable for display, especially such a program for use on the World Wide Web.

Chapter 12: Sources of Information

12.1. Websites and Web Pages

- https://en.wikipedia.org/wiki/JavaFX
- http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html
- http://www.dictionary.com/browse/web--browser?s=t

12.2. Books

Dea, Carl, Gerret Grunwald, Jose Pereda, Sean Phillips, and Mark Heckler. *Javafx 9 by Example*. Apress, 2017.

Ludin, Stephen, and Javier Garza. *Learning HTTP/2: a practical guide for beginners*. Boston, MA: OReilly, 2017.

Schildt, Herbert. *Java: A Beginners Guide, Sixth Edition*. Oracle Press, 2014.

Schildt, Herbert. *Java: The Complete Reference 9th Edition*. New York, NY: McGraw-Hill Companies., 2015.

Chapter 13: Index

Southwest Minnesota State University
Fall 2017 Semester

Class: COMP 425-01 "Software Engineering"
Professor: Kourosh Mortezapour

Team Project: Icewolf web browser
Team Members: Nikolay Ivanov, Gregory Bowen, Dylan Parsons

DESIGN DOCUMENT

Updated: October 23, 2017

Table of Contents

Chapter 1: Introduction

This document describes the preliminary and detailed design of Icewolf, a cross-platform lightweight web-browser based on JavaFX framework. The browser has a developer-ready well-organized source code and is licensed under a permissive open-source license[1]. The web-browser has a modular organization that would allow developers to easily implement new features for the browser without utilizing a complicated (and sometimes unsafe) plugin infrastructure.

The preliminary design of the browser include the External Design Specifications and the External Design Specifications, whereas the detailed design includes, but not limited to, the Detailed Data Flow Diagrams, Detailed Design Specifications, Pseudocodes and Data Structures. The developers, managers and testers of the Icewolf project put a significant effort into keeping the design of the browser clear, unambiguous, efficient and teamwork-oriented.

Chapter 2: Preliminary Design

2.1. External Design Specifications

External design specifications of Icewolf include the following items:

- Main Window
- Menu Bar
- Toolbar
- Tab Bar
- Web View
- Settings Dialog
- Bookmark Manager

The rest of *Section 2.1* is devoted to explanation of the purpose and functionality of the items above. *Section 3.2* provides more details about the elements of the external design of Icewolf web browser.

2.1.1. Main Window

The Main Window of Icewolf web-browser is designed to be a visual and logical top-level container of the program. All user-oriented elements of the browser are supposed to be accessible through the Main Window or its child elements. *Figure 1* shows how the Main Window of the browser looks like. It contains the Main Menu, Tab Bar, Tool Bar and Web View for immediate access by the user. The items that are not visible immediately in the Main Window, the Bookmark Manager and the Settings Dialog, are still to be accessed through the Main Window using the Main Menu.

---

1 https://www.apache.org/licenses/LICENSE-2.0

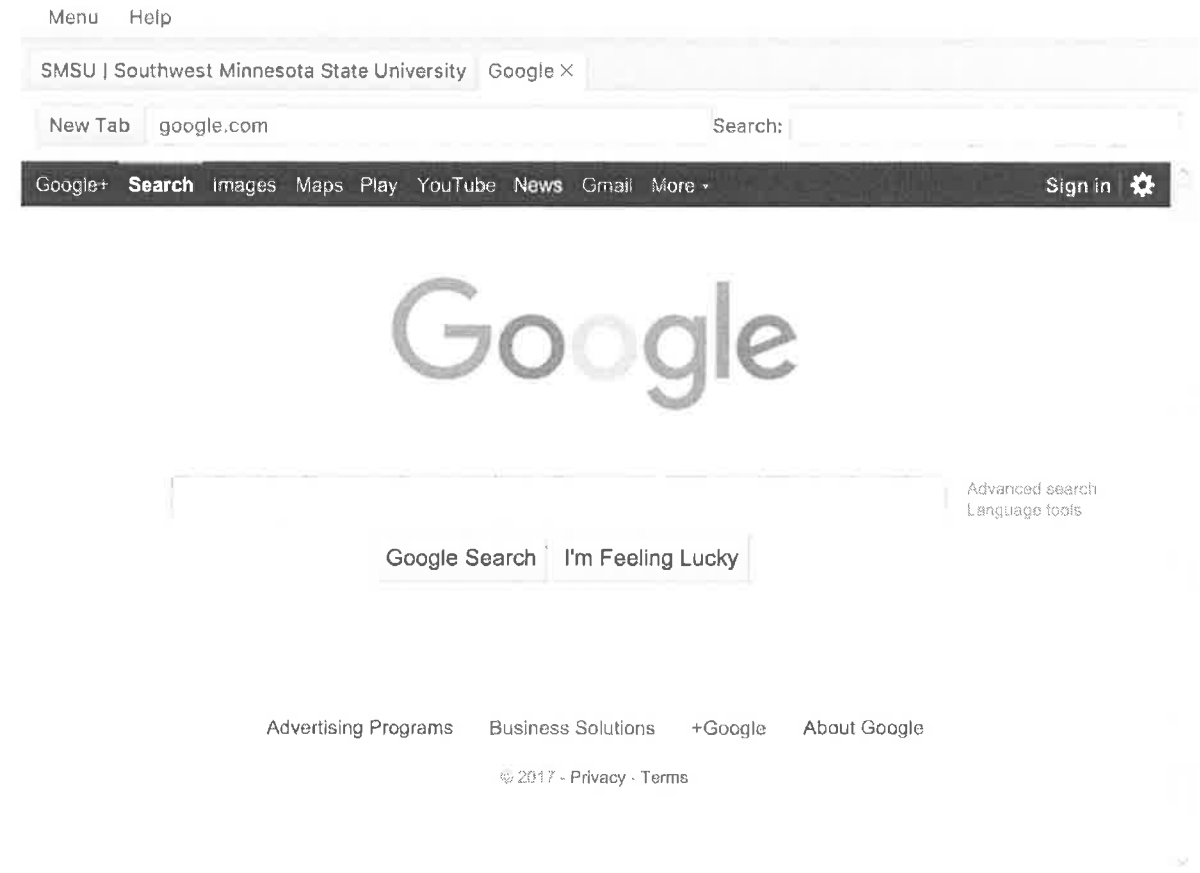


Figure 1. Main Window of Icewolf web-browser with Google.com website loaded.

Chapter 3.1.1. Subsystems and Managers of this document provides more detailed information about the Main Window of the browser. Also, *Chapter 2.2. Internal Design Specifications* discusses the underlying developer-faced code design of the Main Window of the browser.

2.1.2. Menu Bar

The Menu Bar of Icewolf web browser is intended to provide access to various functions of the browser. *Figure 2* shows the Menu Bar's location in the browser's Main Window.

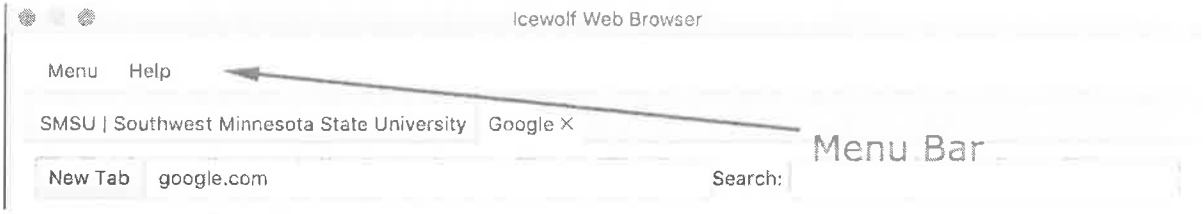


Figure 2. Menu Bar of Icewolf web browser.

Menu Bar and its menus and sub-menus are discussed in detail in *Chapter 3.1.1. Subsystems and Managers* of this document. Also, *Chapter 2.2. Internal Design Specifications* discusses the underlying developer-faced code design of the Menu Bar of the browser.

2.1.3. Toolbar

The Toolbar of Icewolf web browser groups together and provides access to the widgets of the browser that are likely to be frequently used: New Tab Button, Address Bar and Search Bar. *Figure 3* shows the top part of the main window of the web browser with Toolbar pointed on.

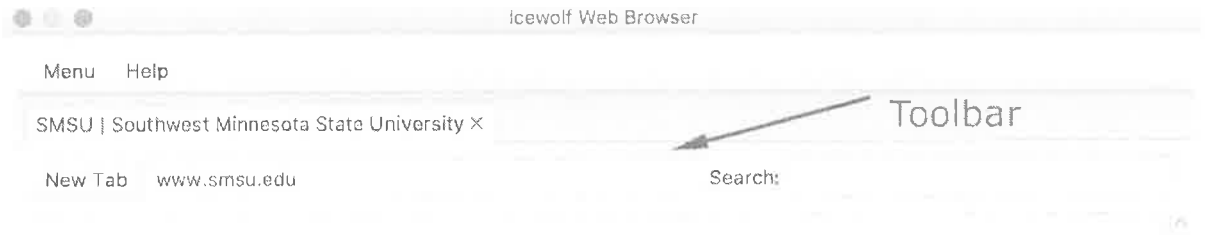


Figure 3. Toolbar of Icewolf web browser.

The sub-elements of Icewolf Toolbar as well as the expected behavior of these elements are discussed in *Section 3.1.1. Subsystems and Managers*.

2.1.4. Tab Bar

The Tab Bar in Icewolf web browser allows to load several web pages in one browser window and switch between them using tabs. *Figure 4* demonstrates the Tab Bar interface within Icewolf web browser's main window.

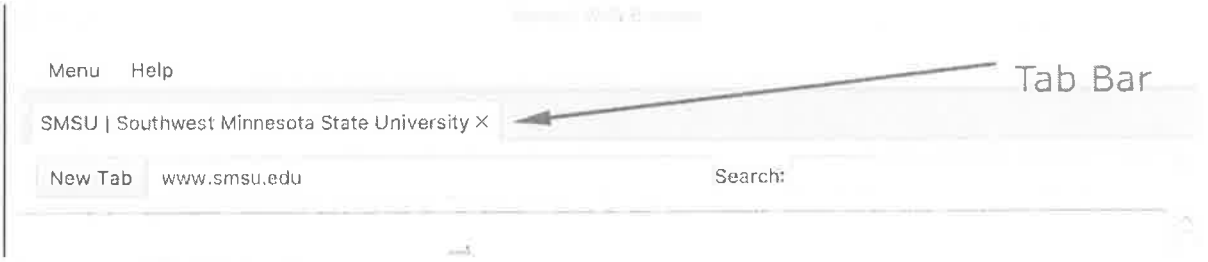


Figure 4. Icewolf Tab Bar interface.

Tab Bar and its multi-threading design are discussed in detail in *Chapter 3.1.1. Subsystems and Managers* of this document. Also, *Chapter 2.2. Internal Design Specifications* discusses the underlying developer-faced code design of the Tab Bar of the browser.

2.1.5. Web View

Web View allows to show web pages as a widget of the browser. *Figure 5* demonstrates the browser's WebView widget bordered by a red rectangle. On the right side of the Web View we can see a scroll bar which allows to scroll pages that are vertical larger than the screen

height. If a page is not resizable and larger than the screen width, a horizontal tab bar will appear on the screen within the Web View.

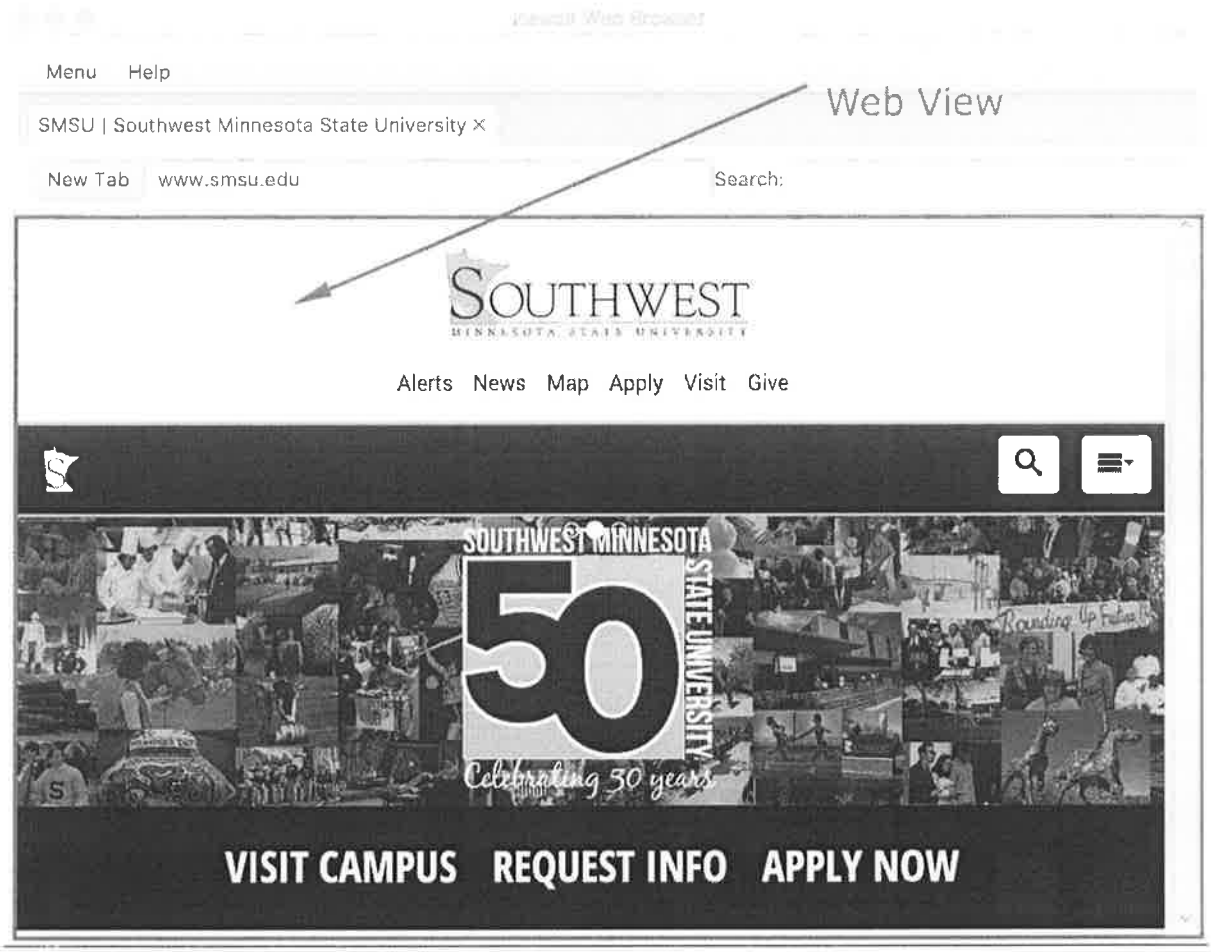
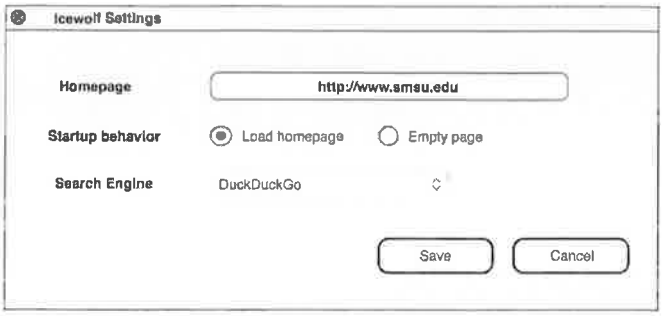


Figure 5. Icewolf web browser Web View interface with SMSU website homepage loaded.

2.1.6. Settings Dialog

The Settings Dialog in Icewolf web browser allows to obtain, modify and store the settings of the browser. The dialog is launched by Menu → Settings menu command. The settings are stored in the browser's SQLite Database which is created and initialized for each user during the initial start of the browser. *Chapter 2.2. Internal Design Specifications* discusses the underlying developer-faced code design of the Settings Dialog of the browser as well as the settings database subsystem of Icewolf web browser. The sub-widgets of Icewolf Settings Dialog as well as the expected behavior of these elements are discussed in *Section 3.1.1. Subsystems and Managers.*



The following is the list of the search engines available to select from the Search Engine drop list:

- DuckDuckGo
- Google
- Bing
- Yandex
- Yahoo!

2.1.7. Bookmark Manager

The bookmark manager allows to create, remove, edit and use bookmarks and classify them into categories. Bookmark manager also allows to create, delete and edit categories. The information is persistently stored in SQLite database that stores individual information for each browser's user. *Figure 6* shows the example visual layout of the Bookmark Manager.

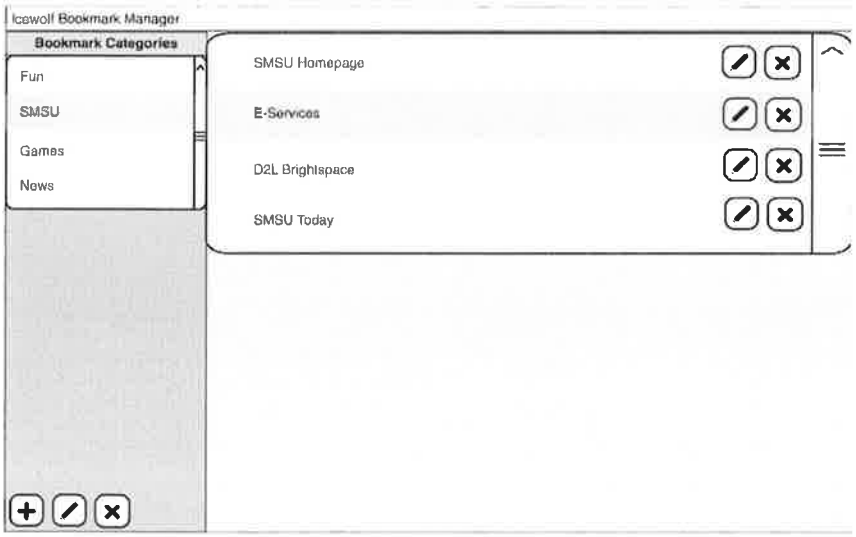


Figure 6. Icewolf Bookmark Manager interface.

Unlike the Settings Dialog, the Bookmark Manager is not opened in a separate window. Instead, when called by Menu → Bookmarks command, it is opened in a new tab. When a user selects a bookmark and double clicks on it, the bookmark manager is substituted by a Web

View attempting to load a web page from the bookmark. *Section 3.1.1. Subsystems and Managers* discusses the design of the Bookmark Manager in detail.

2.2. Internal Design Specifications

Internal design specifications of Icewolf include the following items:

- Java classes
- JavaFX standard widgets
- JavaFX custom widgets
- Settings, preferences, and database management subsystem.

The rest of *Section 2.2* is devoted to explanation of the purpose and functionality of the items above. *Sections 3.1 and 3.2* provides more details about the elements of the internal design of Icewolf web browser.

2.2.1. Java Classes

Icewolf web browser implements the following Java classes:

- IWMenuBar -- implements the Menu Bar;
- IWMenuTab -- implements the Tab Bar;
- IWPropertyHelper -- implements developer-customizable property routines;
- IWSampleModule -- implements a Sample Module for open-source developers to use;
- IWSearchBox -- implements the Search Box;
- IWSettingsModule -- implements the Settings Dialog;
- IWURLField -- implements the Address Bar routines;
- Icewolf -- the main class of the browser;
- IWDatabaseHelper -- implements SQLite 3 database routines;
- IWSettingsHelper -- implements the functionality behind the Settings Dialog.

These classes are discussed in detail in *Section 3.1. Detailed Data Flow Diagrams*.

2.2.2. JavaFX Standard Widgets

Icewolf web browser implements the following JavaFX standard widgets[2]:

- MenuBar
- MenuItem
- TabBar
- Tab
- Button

---

2 Some of these widgets are used indirectly by providing its full functionality as a base class inherited by a custom widget's class. Oracle recommends using this approach to achieve better design of JavaFX GUI interfaces: http://www.oracle.com/technetwork/articles/java/javafx-productivity-2345000.html.

- TextField
- RadioButton
- WebView
- Label
- ComboBox

These widgets are discussed in detail in *Section 3.1. Detailed Data Flow Diagrams* and *Section Error: Reference source not found*.

2.2.3. JavaFX Custom Widgets

Along with standard JavaFX widgets, Icewolf web browser implements custom widgets by extending standard widgets' base classes, adding new functionality, and adding new features. The following are custom widgets implemented in Icewolf web browser:

- **IWInternetTab** extends Tab;
- **IWMenuBar** extends MenuBar
- **IWSearchBox** extends TextField
- **IWURLField** extends TextField

These widgets are discussed in detail in *Section 3.1. Detailed Data Flow Diagrams* and *Section Error: Reference source not found*.

2.2.4. Settings, Preferences and Database Management Subsystem

Icewolf web browser includes three special subsystems: Settings, Preferences and Database Management ones. These subsystems allow to personify the web browser for each individual user.

Settings can be changed by a user using the Settings Dialog. They can be applied dynamically without rebuilding or restarting the program. Settings Subsystem uses the Database Subsystem to store data for each individual user.

Preferences Subsystem is less dynamic and allows to accommodate the browser for certain user's or developer's needs through changing the Java preferences file. This Subsystem implies rebuild of the project and restart of the program each change made in the preferences. With that being said, the Preferences subsystem is not suitable for frequently adjustable changes. However, for the changes that are likely to be made rarely or once, the Preferences subsystem provides a significant performance boost in comparison with a slower-by-design dynamic Settings Subsystem.

The database subsystem allows to store Settings and Bookmarks in a database driven by SQLite 3 engine.

Chapter 3: Detailed Design

3.1. Detailed Data Flow Diagrams

3.1.1. Subsystems and Managers

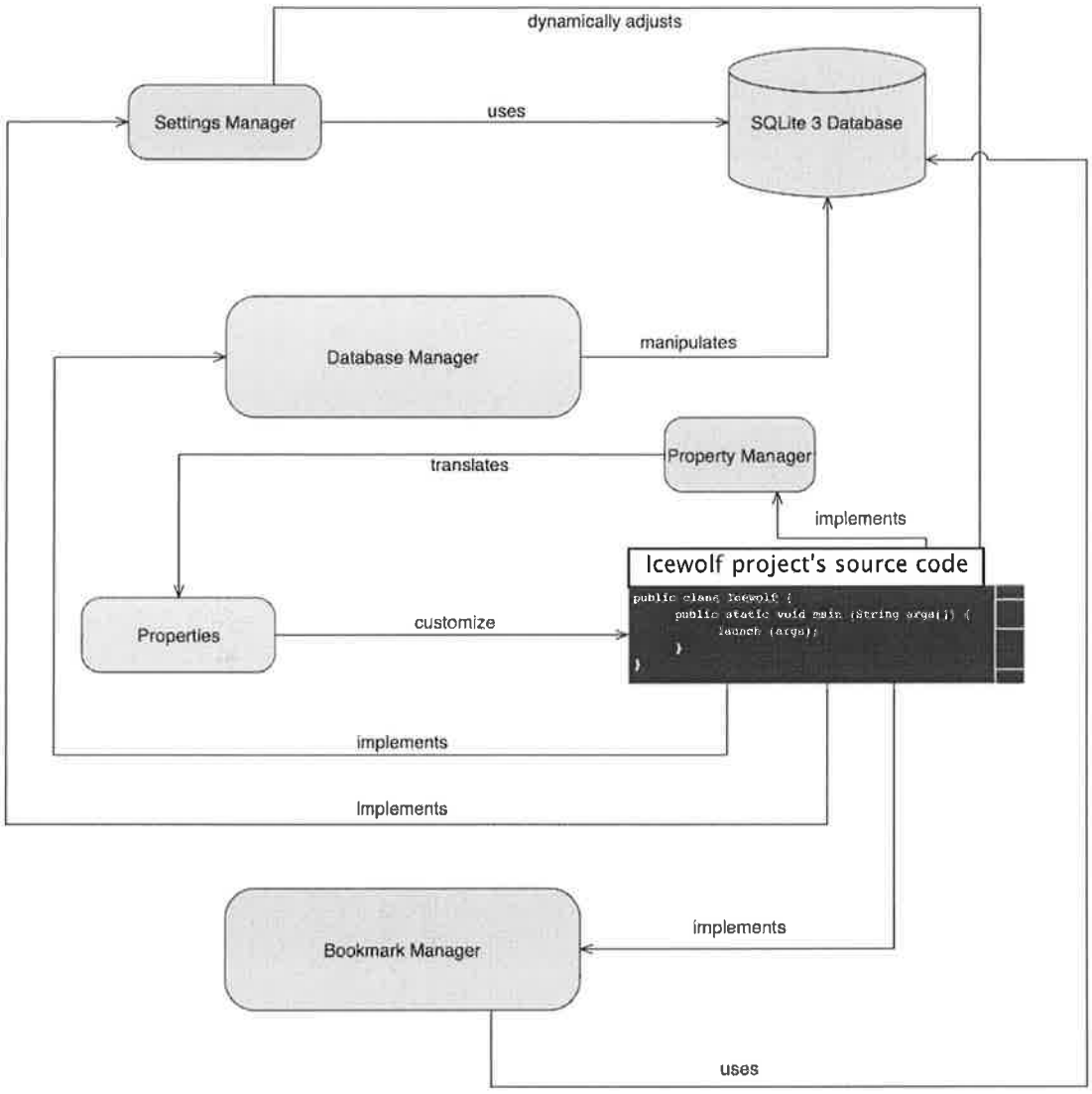


Figure 7. Subsystems and Managers of Icewolf web browser.

3.1.2. SQLite Database

Figure 8 shows the design of the database of Icewolf browser. As we can see, the database has three tables, two of which are related with a foreign key.

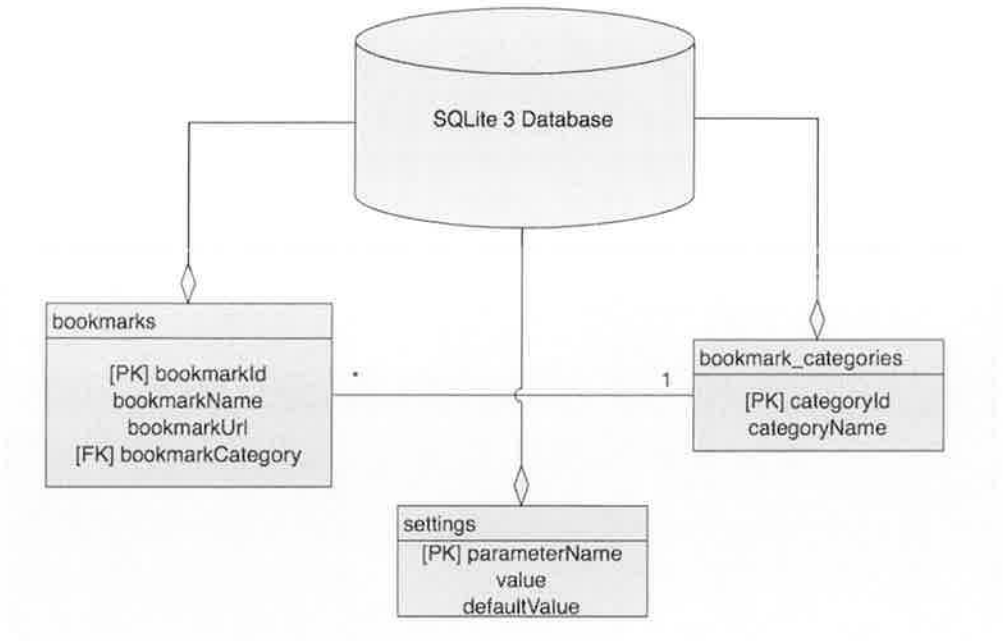


Figure 8. Icewolf web browser SQLite Database design

As it was said in *Section 2.2.4. Settings, Preferences and Database Management Subsystem*, the database subsystem allows to store Settings and Bookmarks in a database driven by SQLite 3 engine. The bookmarks are stored in **bookmarks** database table. This table is related to **bookmark_categories** table which stores categories of bookmarks. The browser's settings are stored in **settings** table of the database; this table has no relationships with other tables.

3.2. Detailed Design Specifications

3.2.1. Main Window

Figure 9 demonstrates the relations between Icewolf's Main Window, other widgets, and items of the browser's source code.

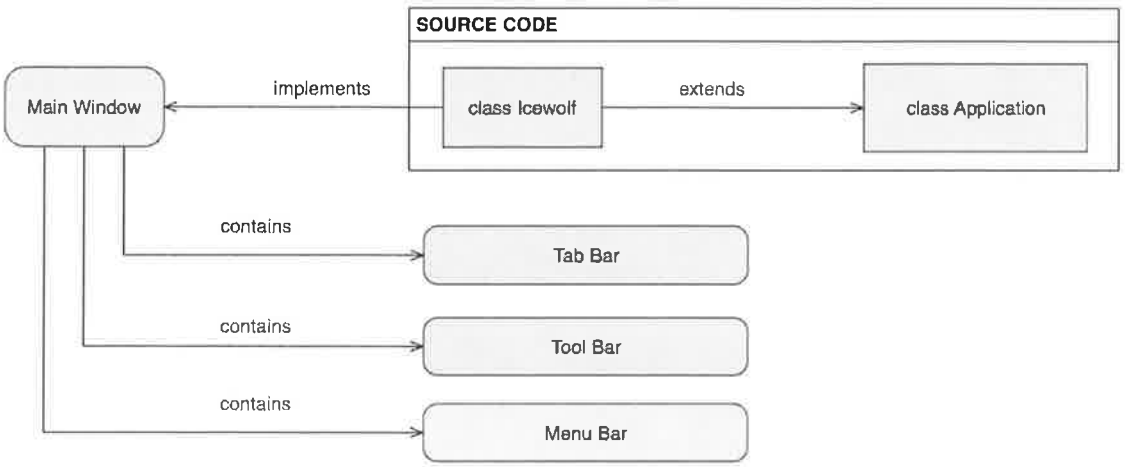


Figure 9. The relations between Icewolf's Main Window, its subwidgets and the source code.

3.2.2. User-triggered Event Map

Table 1 shows the relations between user-triggered events and inputs and the reactions of the web browser on these events.

Location	Event	Action
Main Menu	Select Menu -> Settings item	Settings Dialog window opens
Main Menu	Select Menu -> Bookmark Manager item	Icewolf Bookmark Manager is launched in a new tab
Main Menu	Select Menu -> Quit item	The browser exits
Main Menu	Select Help -> Manual	Icewolf user documentation web-page is opened in a new tab
Main Menu	Select Help -> About	An information dialog pops up providing basic information about the program, licensing details and contact information
Main Menu	Select Menu -> Add Bookmark item	A dialog appears asking to add a bookmark with the title of the current page as a proposed name of the bookmark and a dropdown list of existing categories.
Tool Bar	URL typed in the Address Bar and <Enter> key pressed	A web page with the provided URL is attempted to be opened in the current tab of the Tab Bar
Tool Bar	Search criterion is typed in the Search Barand <Enter>	A default search engine, Specified in the Settings, is launched in a new tab

	key pressed	with the search criterion's search results.
Tool Bar	New Tab button is clicked	A new tab is opened with no page loaded (homepage is not used for new tabs when the browser is already running).
Settings Manager	Homepage text input is changed	The setting is not stored in the database until Save button is clicked
Settings Manager	Startup Behavior radio button is changed	The setting is not stored in the database until Save button is clicked
Settings Manager	Search Engine is selected from the dropdown list.	The setting is not stored in the database until Save button is clicked
Settings Manager	Cancel button clicked	Any changes in settings are discarded and the Settings Dialog is closed
Settings Manager	Save button clicked	Changes are saved to the database, but the window of the Settings Dialog remains open.
Bookmarks Manager	Add button is clicked at the Categories panel	A new row appears inviting to type the name of a new bookmark category. Once user types it and presses <Enter>, the change immediately goes into the database.
Bookmark Manager	Edit button[3] is clicked at the Categories panel.	The currently selected category becomes editable within the same window and the same placeholder. After the user edits the name and presses <Enter>, the changes immediately query into the database.
Bookmark Manager	Delete button[4] is clicked in the Categories panel.	The current category will be deleted and the cascade query will be triggered removing all the bookmarks within the category[5]. The changes will be made to the

3 The button will not be active until one of the categories selected.
4 The button will be inactive until a category is selected.
5 Because bookmarks don't belong to the category of extremely precious information, the delete confirmation dialog would be redundant in this case. Future developers and/or fork-developers, however, can introduce a confirmation dialog, if needed.

		database immediately.
Bookmark Manager	Edit button is clicked in the Bookmark pane next to a bookmark	A popup dialog appears allowing to change name and/or category of the bookmark. The changed are queried into the database immediately after clicking Confirm button. Cancel button is also provided.
Bookmark Manager	Delete button is clicked in the Bookmark pane next to a bookmark	The bookmark is deleted right away and the changes are queried into the database.

3.2.3. Menu Bar

Figure 10 demonstrates the relations between Icewolf's Menu Bar, other widgets, and items of the browser's source code.

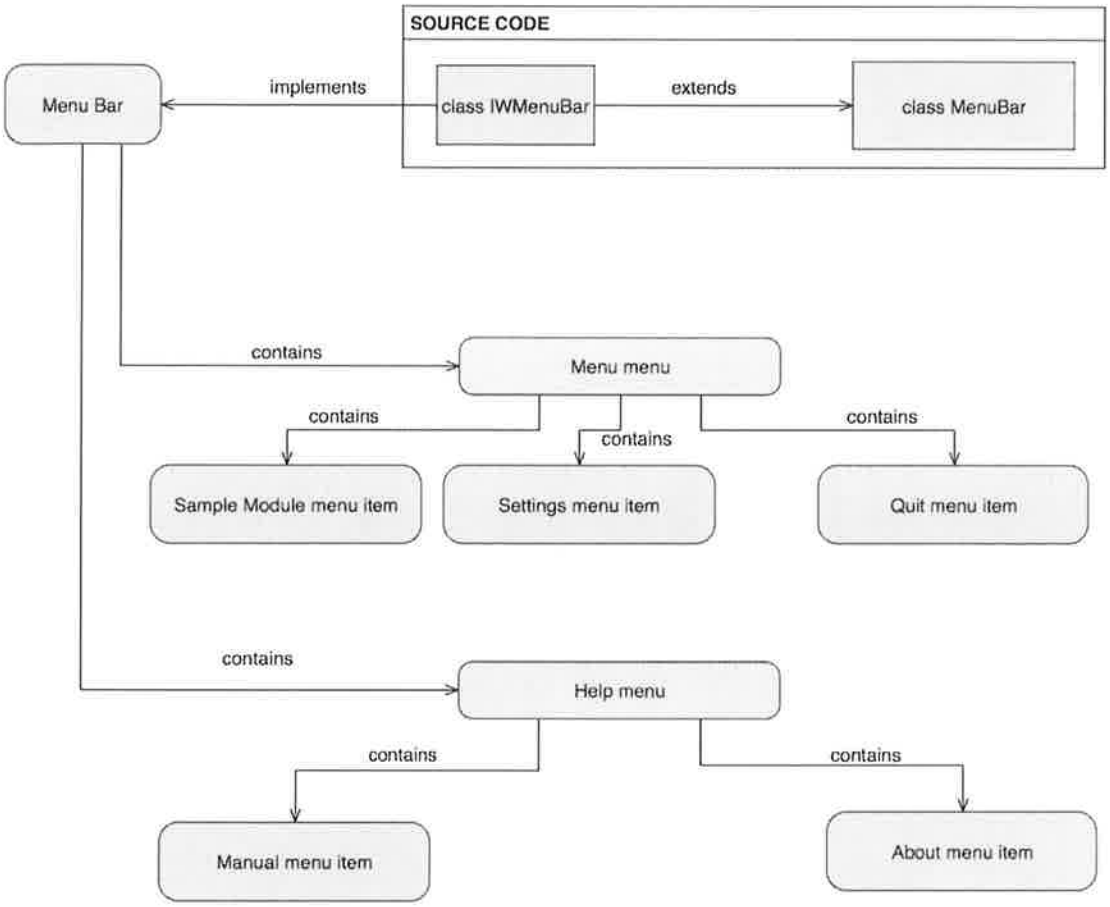


Figure 10. The relations between Icewolf's Menu Bar, its subwidgets and the source code.

Glossary

JavaFX: JavaFX is a software platform for creating and delivering desktop applications that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS[6].

Raspberry Pi: The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

SQLite: SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

Web Browser: A software program that allows the user to find and read encoded documents in a form suitable for display, especially such a program for use on the World Wide Web

---

6 More information here:
https://en.wikipedia.org/wiki/JavaFX
http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html

Sources of Information

12.1. Websites and Web Pages

- https://en.wikipedia.org/wiki/JavaFX
- http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html
- http://www.dictionary.com/browse/web--browser?s=t
- http://www.oracle.com/technetwork/articles/java/javafx-productivity-2345000.html

12.2. Books

Dea, Carl, Gerret Grunwald, Jose Pereda, Sean Phillips, and Mark Heckler. *Javafx 9 by Example*. Apress, 2017.

Ludin, Stephen, and Javier Garza. *Learning HTTP/2: a practical guide for beginners*. Boston, MA: OReilly, 2017.

Schildt, Herbert. *Java: A Beginners Guide, Sixth Edition*. Oracle Press, 2014.

Schildt, Herbert. *Java: The Complete Reference 9th Edition*. New York, NY: McGraw-Hill Companies., 2015.

Southwest Minnesota State University
Fall 2017 Semester

Class: COMP 425-01 "Software Engineering"
Professor: Kourosh Mortezapour

Team Project: Icewolf web browser
Team Members: Nikolay Ivanov, Gregory Bowen, Dylan Parsons

PROJECT LEGACY

Updated: November 28, 2017

Table of Contents

Project Description

Icewolf is a project of creating a cross-platform lightweight web-browser based on JavaFX framework. The browser will have developer-ready well-organized source code and will be licensed under a permissive open-source license. The web-browser will have a modular organization that would allow developers to easily implement new features for the browser without utilizing a complicated (and sometimes unsafe) plugin infrastructure.

Initial Expectations

The following is the list of minimal features that are initially expected to be present in the browser:

- GUI Interface
- Cross-platform: Linux/Windows/Mac
- Full HTML5 Support
- Full CSS3 Support
- Full JavaScript/ECMA Support
- Modular structure
- Menu bar
- Universal address bar
- HTTP/HTTPS-protocol auto-recognition

Current Status of the Project

Post-development (11/20 – 12/6) – Implementing Tabs, Implementing Search Bar, Bug Fixes (see the *Opportunity Study* document for more information).

Remaining Areas of Concern

The previous issues and concerns have been solved. No new issues reported by the team members.

Solution Strategy

In order to implement the project, our team is going to use the Agile and Extreme Programming methodologies with Scrum meetings, 2-week iterations, comprehensive multi-faceted unit testing, and inter-team role distribution.

The following is the Agile Development Calendar for the project:

STAGE	DATES	ACTIVITIES	EST. HOURS OF WORK
Pre-development	8/30 – 9/11	• Ideas Brainstorming • Introductory Meeting • Pre-planning Meeting • Opportunity Study	25 h.

Iteration 1: Core Infrastructure	9/11 – 9/25	• Iteration 1 Planning Meeting [1] • Iteration 1 Scrum Meeting 1 [1/8] • Iteration 1 Scrum Meeting 2 [1/8] • Iteration 1 Scrum Meeting 3 [1/8] • Iteration 1 Scrum Meeting 4 [1/8] • Iteration 1 Scrum Meeting 5 [1/8] • Iteration 1 Scrum Meeting 6 [1/8] • Iteration 1 Scrum Meeting 7 [1/8] • Iteration 1 Scrum Meeting 8 [1/8] • Iteration 1 Individual Development [4] • Iteration 1 Pair Development [3] • Iteration 1 Testing [1] • Iteration 1 Documentation [1] • Iteration 1 Paperwork and Storyboard Maintenance [1] • Iteration 1 Release [1]	42 h.
Iteration 2: Tab Bar and Search Field Implementation	9/25 – 10/9	• Iteration 2 Planning Meeting • Iteration 2 Scrum Meeting 1 • Iteration 2 Scrum Meeting 2 • Iteration 2 Scrum Meeting 3 • Iteration 2 Scrum Meeting 4 • Iteration 2 Scrum Meeting 5 • Iteration 2 Scrum Meeting 6 • Iteration 2 Scrum Meeting 7 • Iteration 2 Scrum Meeting 8 • Iteration 2 Bug Fixes [1] • Iteration 2 Individual Development [3] • Iteration 2 Pair Development • Iteration 2 Testing • Iteration 2 Documentation • Iteration 2 Paperwork and Storyboard Maintenance • Iteration 2 Release	47 h.
Iteration 3: Menu Bar Implementation and Performance Improvements	10/9 – 10/23	• Iteration 3 Planning Meeting • Iteration 3 Scrum Meeting 1 • Iteration 3 Scrum Meeting 2 • Iteration 3 Scrum Meeting 3 • Iteration 3 Scrum Meeting 4 • Iteration 3 Scrum Meeting 5 • Iteration 3 Scrum Meeting 6	44 h.

		• Iteration 3 Scrum Meeting 7 • Iteration 3 Scrum Meeting 8 • Iteration 3 Bug Fixes • Iteration 3 Individual Development • Iteration 3 Pair Development • Iteration 3 Testing • Iteration 3 Documentation • Iteration 3 Paperwork and Storyboard Maintenance • Iteration 3 Release	
Iteration 4: Database Controller and Download Manager Implementation	10/23 – 11/6	• Iteration 4 Planning Meeting • Iteration 4 Scrum Meeting 1 • Iteration 4 Scrum Meeting 2 • Iteration 4 Scrum Meeting 3 • Iteration 4 Scrum Meeting 4 • Iteration 4 Scrum Meeting 5 • Iteration 4 Scrum Meeting 6 • Iteration 4 Scrum Meeting 7 • Iteration 4 Scrum Meeting 8 • Iteration 4 Bug Fixes • Iteration 4 Individual Development • Iteration 4 Pair Development • Iteration 4 Testing • Iteration 4 Documentation • Iteration 4 Paperwork and Storyboard Maintenance • Iteration 4 Release	43 h.
Iteration 5: Settings and Bookmarks	11/6 – 11/20	• Iteration 5 Planning Meeting • Iteration 5 Scrum Meeting 1 • Iteration 5 Scrum Meeting 2 • Iteration 5 Scrum Meeting 3 • Iteration 5 Scrum Meeting 4 • Iteration 5 Scrum Meeting 5 • Iteration 5 Scrum Meeting 6 • Iteration 5 Scrum Meeting 7 • Iteration 5 Scrum Meeting 8 • Iteration 5 Bug Fixes • Iteration 5 Individual Development • Iteration 5 Pair Development	49 h.

		• Iteration 5 Testing • Iteration 5 Documentation • Iteration 5 Paperwork and Storyboard Maintenance • Iteration 5 Release	
Post-development	11/20 – 12/6		37 h.
TOTAL:			**287.5 h.**

Activities and Time Logs

Activity	Nick	Gregory	Dylan
Ideas Brainstorming	2.5 h.	2.5 h.	2.5 h.
Introductory Meeting	0.5 h.	0.5 h.	0.5 h.
Pre-planning Meeting	0.5 h.	0.5 h.	0.5 h.
Opportunity Study	2 h.	0.5 h.	0.5 h.
Individual Development	43 h.	56 h.	49 h.
Pair Programming	12.5 h.	12.5 h.	12.5 h.
Planning Meetings	7.5 h.	7.5 h.	7.5 h.
Standup/Scrum Meetings	4.5 h.	4.5 h.	4.5 h.
Story-related paperwork	3 h.	1 h.	3 h.
Project Legacy Paperwork	3.5 h.	2.5 h.	1.5 h.
Bug fixing	6 h.	8 h.	3 h.
Running Unit Test and Release Management	5 h.	1 h.	2 h.
Documentation Writing	7.5 h.	2.5 h.	3 h.
TOTAL:	98.0	99.5	90.0

Technical Lessons Learned

While working on the project, we have learned the following technical lessons:

- JavaFX has multiple ways of implementing the same interface, ***but only one way***, suggested by Oracle Inc., represents a truly good GUI interface design, which allows future developers to continue working on the project.
- Although Unit tests are considered to be a supplemental part of coding in Extreme Programming methodology, they still consume *at least one quarter* of all the implementation time.

Managerial Lessons Learned

- More code introduces more bugs, which should be considered while planning for subsequent iterations.

- Although Agile development model reduces the overall stress and workload by splitting goals into small tasks and getting rid of unnecessary activities, it requires a good time management and perfect discipline from all team members (attending meetings on time and meeting the intermediate due dates).

Recommendations to Future Projects

After finishing the project, our team has come up with the following recommendations for future projects:

- Selecting the topic: a) spend some time researching opportunities; b) be realistic.
- Selecting the language: choose the language that has a good cross-platform IDE (Java/NetBeans).
- Selecting team members: it is good to form a team of members strong in ***different*** areas of expertise.
- Selecting the time: Agile time-management works well not only in the corporate environment but also in a college project setting.