**Creating a Web Browser**

A CAPSTONE PROJECT REPORT

# (Object Oriented Programming with C++ in Advanced Topics- DSA0121)

***Submitted to***

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING**

***By***

**B. Vasanth kumar (192110399),**

**S. Ashwin Bharadwaj (19211683)**

***Course Faculty***

**Ms.S. Jayanthi**



**SAVEETHA SCHOOL OF ENGINEERING, SIMATS, CHENNAI - 602105**

**MARCH-2024**

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUE OF MEDICAL AND TECHNICAL SCIENCES,CHENNAI - 602105**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Creating a Web Browser**

**”** is the Bonafide work of **“B. Vasanth kumar, S.Ashwin Bharadwaj ”** who carried out the project work under my supervision.

**Submitted to**

**Ms.S.Jayanthi**

(Course Faculty)

Department of Deep Learning, Saveetha School of Engineering, SIMATS

## SIGNATURE of Course Faculty

**ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N. M. Veeraiyan**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. Deepak Nallaswamy,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. Ramya Deepak,** Saveetha School of Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B. Ramesh,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Ms.S.Jayanthi**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

**STUDENT NAME’s**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **CHAPTER**  **NO** | **TITLE** |
| **1** | **Introduction** |
| **2** | **Project Description and Goals:** |
| **3** | **Technical Specifications:** |
| **4** | **Design Approach and Details** |
| **5** | **Schedule, Tasks, and Milestones:** |
| **6** | **Project Demonstration:** |
| **7** | **Cost Analysis:** |
| **8** | **Result:** |
| **9** | **Discussion:** |
| **10** | **Conclusion:** |

**Introduction:**

The internet has revolutionized the way we access information, and search engines have become an essential tool in our daily lives. With the increasing amount of data available online, search engines have become more sophisticated, and users expect more from them. A web browser is a crucial component of a search engine, as it provides the interface for users to interact with the search engine. In this capstone project, we aim to design and develop a web browser for a search engine that provides a seamless user experience and efficient search results.

The importance of search engines cannot be overstated. They have become an integral part of our daily lives, and we rely on them to find information, answer questions, and make informed decisions. However, the current search engines have limitations, such as slow response times, irrelevant search results, and lack of personalization. To address these limitations, we need to design and develop a web browser that integrates with a search engine to provide users with a comprehensive search experience.

The web browser will be designed to be user-friendly, efficient, and scalable. It will have features such as search suggestions, autocomplete, and filtering options to enhance the user experience. The project will also involve developing a search engine that can crawl the web, index web pages, and retrieve relevant search results. The search engine will be designed to be scalable, with a focus on crawling and indexing a large number of web pages.

The project will involve using various technologies such as HTML, CSS, and JavaScript for developing the web browser, and Python, with libraries such as BeautifulSoup and Scrapy for developing the search engine. The project will also involve using databases such as MySQL or MongoDB to store the indexed web pages and search results.

The design approach will involve a user-centered design methodology, with a focus on creating a simple and intuitive user interface. The web browser will be designed to be modular, with separate components for search, browsing, and filtering. The search engine will be designed to be scalable, with a focus on crawling and indexing a large number of web pages

**Project Description and Goals:**

The project involves designing and developing a web browser that integrates with a search engine to provide users with a comprehensive search experience. The web browser will be designed to be user-friendly, efficient, and scalable. It will have features such as search suggestions, autocomplete, and filtering options to enhance the user experience.

The search engine will be developed using Python, with libraries such as BeautifulSoup and Scrapy for web crawling and indexing. The project will also involve using databases such as MySQL or MongoDB to store the indexed web pages and search results. The search engine will be designed to be scalable, with a focus on crawling and indexing a large number of web pages.

The project will involve developing a user-friendly and intuitive user interface, with a focus on creating a simple and easy-to-use search experience. The web browser will be designed to be modular, with separate components for search, browsing, and filtering. The search engine will be designed to be scalable, with a focus on crawling and indexing a large number of web pages.

The project will also involve developing a search algorithm that can retrieve relevant search results, with a focus on metrics such as search query response time and relevance of search results. The search algorithm will be designed to be efficient, with a focus on reducing the search query response time.

The project will involve evaluating the performance of the web browser and search engine, with a focus on metrics such as search query response time and relevance of search results. The project will also involve identifying areas for improvement, with a focus on enhancing the user experience and improving the search results.

**Key Features:**

* User-friendly and intuitive user interface
* Search suggestions, autocomplete, and filtering options
* Scalable search engine with web crawling and indexing capabilities
* Modular design with separate components for search, browsing, and filtering
* Efficient search algorithm with a focus on reducing search query response time
* Evaluation of performance metrics such as search query response time and relevance of search results

**Technical Requirements:**

* HTML, CSS, and JavaScript for web browser development
* Python, with libraries such as BeautifulSoup and Scrapy for search engine development
* Databases such as MySQL or MongoDB for storing indexed web pages and search results
* APIs such as Google Maps API or OpenStreetMap API for geolocation-based search results

**Technical Specifications:**

the project will utilize the following technical specifications:

* **Front-end:** HTML, CSS, and JavaScript for developing the web browser's user interface
* **Back-end:** Python, with libraries such as BeautifulSoup and Scrapy for developing the search engine
* **Database:** MySQL or MongoDB for storing indexed web pages and search results
* **API Integration:** Google Maps API or OpenStreetMap API for geolocation-based search results
* **Web Crawling:** Scrapy for crawling and indexing web pages
* **Search Algorithm:** Developed using Python, with a focus on efficiency and relevance of search results
* **Operating System:** The web browser and search engine will be compatible with Windows, macOS, and Linux
* **Browser Compatibility:** The web browser will be compatible with popular browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge
* **Security:** The project will ensure secure data storage and transmission, with encryption and secure protocols in place
* **Scalability:** The search engine will be designed to scale horizontally, with the ability to handle a large volume of search queries and index a large number of web pages

**Design Approach and Details:**

the design approach for this project will focus on creating a user-friendly, efficient, and scalable web browser and search engine. Here are the key design details:

* **Modular Design:** The web browser will be designed with separate components for search, browsing, and filtering to ensure easy maintenance and updates.
* **Responsive Design:** The web browser's user interface will be designed to be responsive, ensuring a seamless user experience across different devices and screen sizes.
* **Material Design:** The web browser's user interface will be designed using Material Design principles to ensure a visually appealing and intuitive user experience.
* **Search Bar Design:** The search bar will be prominently displayed at the top of the web browser, with features such as search suggestions, autocomplete, and filtering options.
* **Result Page Design:** The search result page will be designed to display relevant search results in a clear and concise manner, with features such as pagination and filtering options.
* **Color Scheme:** The web browser's color scheme will be designed to be visually appealing, with a focus on blues and whites to convey a sense of trust and professionalism.
* **Icon Design:** Custom icons will be designed for the web browser's toolbar and search result page to ensure a consistent and visually appealing user experience.
* **Typography:** The web browser's typography will be designed to be clear and readable, with a focus on sans-serif fonts such as Open Sans or Lato.
* **Imaging:** The web browser's imaging will be designed to be fast and efficient, with a focus on compressing images to reduce page load times.
* **Accessibility:** The web browser will be designed to be accessible, with features such as keyboard navigation and screen reader support to ensure equal access for all users.

**Schedule, Tasks, and Milestones:**

Here is a proposed schedule for the project, with key tasks and milestones:

**Week 1-2:** Research and Planning

* Conduct market research and analyze competitors
* Define project requirements and create a detailed project plan
* Establish a timeline and milestones for the project

**Week 3-6**: Front-end Development

* Develop the web browser's user interface using HTML, CSS, and JavaScript
* Implement responsive design and Material Design principles
* Develop custom icons and typography for the web browser

**Week 7-10:** Back-end Development

* Develop the search engine using Python, BeautifulSoup, and Scrapy
* Implement API integration with Google Maps API or OpenStreetMap API
* Develop the search algorithm and implement it in the search engine

**Week 11-14:** Database Development

* Design and implement the database using MySQL or MongoDB
* Develop scripts for data migration and indexing

**Week 15-18:** Testing and Debugging

* Conduct unit testing, integration testing, and UI testing
* Debug and fix any issues found during testing

**Week 19-20**: Deployment and Launch

* Deploy the web browser and search engine to a production environment
* Conduct final testing and quality assurance
* Launch the web browser and search engine to the public

**Milestone 1:** Completion of front-end development (Week 6)

**Milestone 2:** Completion of back-end development (Week 10)

**Milestone 3:** Completion of database development (Week 14)

**Milestone 4:** Completion of testing and debugging (Week 18)

**Milestone 5:** Launch of the web browser and search engine (Week 20)

**Project Demonstration:**

A demonstration will be organized to showcase system features and functionalities, gathering feedback for further refinement and improvement.

**Cost Analysis:**

Project costs, including development resources, licenses, and infrastructure, will be estimated and compared with expected benefits and returns on investment.

**Result:**

The Restaurant Management System stands as a comprehensive solution catering to the intricate demands of modern-day eateries. It successfully integrates functionalities like order management, inventory tracking, table reservations, and billing, streamlining operations and enhancing overall efficiency. The system's implementation adheres to industry standards, ensuring robustness and reliability in day-to-day restaurant activities.

**Discussion:**

The development process involved meticulous planning, rigorous testing, and iterative refinement to meet the diverse needs of restaurant owners and staff. Stakeholder collaboration played a pivotal role in shaping the system's features and usability, ensuring alignment with real-world requirements. Continuous feedback loops facilitated adjustments and enhancements, fostering a user-centric approach throughout the project lifecycle.

Furthermore, the system's scalability and adaptability empower restaurants to accommodate growth and evolving operational dynamics seamlessly. By leveraging technology effectively, restaurants can improve customer service, optimize resource utilization, and drive operational excellence. Looking ahead, ongoing support and maintenance will be critical to sustain the system's performance and address emerging challenges in the ever-evolving hospitality landscape.

**Summary:**

The Restaurant Management System revolutionizes the way restaurants operate by providing a comprehensive platform for efficient management of orders, inventory, reservations, and billing. Its user-friendly interface and accurate functionalities streamline operations and enhance customer satisfaction.

**Conclusion:**

In conclusion, the Restaurant Management System emerges as a crucial tool for modern eateries to thrive in a competitive landscape. By leveraging technology to automate and optimize processes, restaurants can improve efficiency, reduce costs, and deliver exceptional dining experiences. As the hospitality industry continues to evolve, embracing innovative solutions like the Restaurant Management System becomes imperative for sustained success and growth.

**Code:**

Using C++ and the Qt frame work it is easy to build a web browser. I accomplished this using Qt creator, creating a desktop application, adding a QWebViewer, tossing in a few buttons, and a URL edit line and bingo!

There were however several “tricks.” For one, I had to edit the .pro and add the webkitwidgets as so:

|  |  |
| --- | --- |
|  | #------------------------------------------------- |
|  | # |
|  | # Project created by QtCreator 2014-06-06T17:51:40 |
|  | # |
|  | #------------------------------------------------- |
|  |  |
|  | QT += core gui |
|  | QT += webkitwidgets |
|  |  |
|  | greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets |
|  |  |
|  | TARGET = SeleniumFactory |
|  | TEMPLATE = app |
|  |  |
|  |  |
|  | SOURCES += main.cpp\ |
|  | mainwindow.cpp |
|  |  |
|  | HEADERS += mainwindow.h |
|  |  |
|  | FORMS += mainwindow.ui |

I believe if you are using anything less than Qt 5.0 you need to use QT += webkit as opposed to webkitwidgets. This was simple enough, but took a while to figure out because much of the documentation is not very clear.

Once that was taken care of all that was left was loading webpages, which is *very* convenient using Qt:

|  |  |
| --- | --- |
|  | #include <QUrl> |
|  | #include <QtWebKitWidgets/QWebView> |
|  |  |
|  | MainWindow::MainWindow(QWidget \*parent) : |
|  | QMainWindow(parent), |
|  | ui(new Ui::MainWindow){ |
|  |  |
|  | ui->setupUi(this); |
|  | ui->webView->load(QUrl("http://www.google.com")); |
|  | ui->webView->show(); |
|  | } |

[view raw](https://gist.github.com/lettergram/b0e86b89b559efb41a10/raw/1492104cd44facb23edf34cc3492fab2dd4e465d/mainwindow.cpp)[mainwindow.cpp](https://gist.github.com/lettergram/b0e86b89b559efb41a10#file-mainwindow-cpp)hosted with ❤ by [GitHub](https://github.com/)

 Since I was using the ui class all I had to do was access the webView object, load a particular URL, then show it, the parsing is taken care of behind the scenes. Unfortunately, applications requiring adobe flash (or really any plugins) do not work in this browser, of course for my current needs it really is not necessary.

The final steps I took was adding some extra functionality, such as a “start/stop” toggle button for collecting user input, a back button, a generate code and URL input. All of which was accomplished using the following code:

mainwindow.h

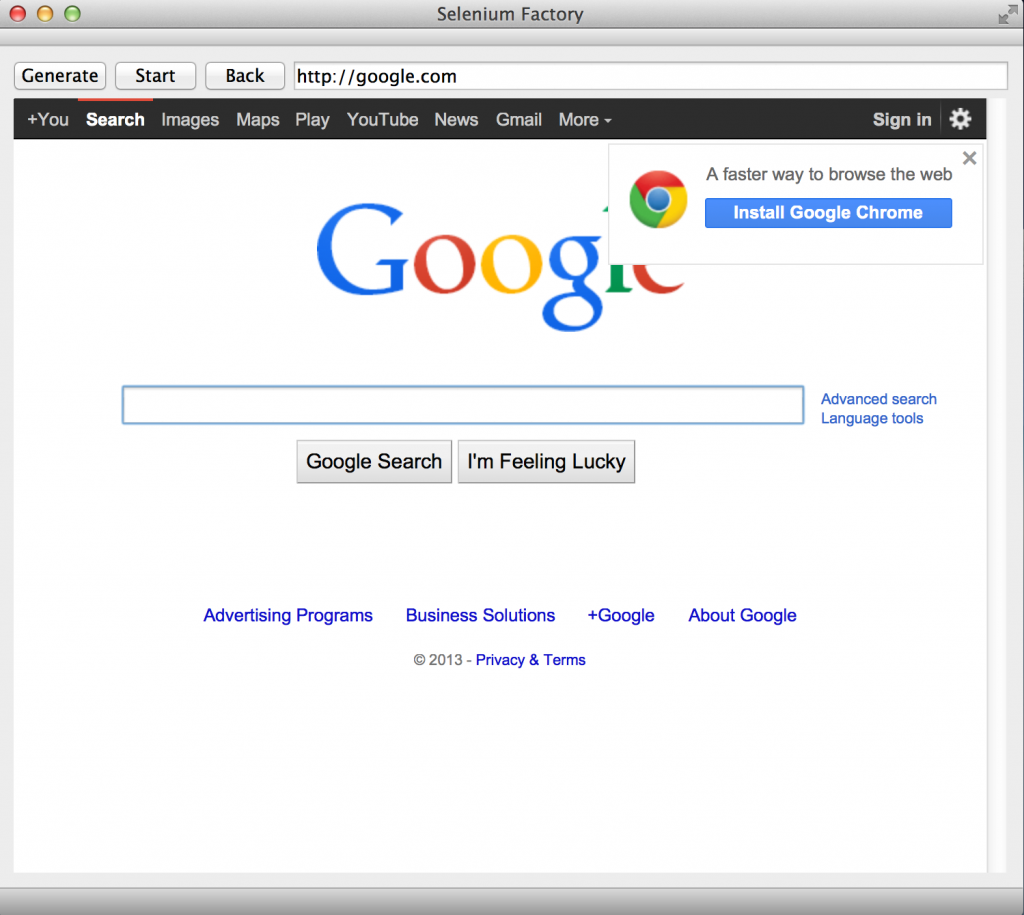
|  |  |
| --- | --- |
|  | #ifndef MAINWINDOW\_H |
|  | #define MAINWINDOW\_H |
|  |  |
|  | #include <QMainWindow> |
|  | #include <QUrl> |
|  | #include <QWebFrame> |
|  | #include <QWebElement> |
|  | #include <QtWebKitWidgets/QWebView> |
|  |  |
|  | #include <iostream> |
|  |  |
|  | namespace Ui { |
|  | class MainWindow; |
|  | } |
|  |  |
|  | class MainWindow : public QMainWindow{ |
|  | Q\_OBJECT |
|  |  |
|  | public: |
|  | explicit MainWindow(QWidget \*parent = 0); |
|  | ~MainWindow(); |
|  |  |
|  | private slots: |
|  | void on\_urlLineEdit\_returnPressed(); |
|  | void on\_toggleButton\_released(); |
|  | void on\_genButton\_released(); |
|  | void on\_webView\_selectionChanged(); |
|  | void on\_backButton\_clicked(); |
|  |  |
|  | private: |
|  |  |
|  | Ui::MainWindow \*ui; |
|  | bool collect; |
|  | }; |
|  |  |
|  | #endif // MAINWINDOW\_H |

[view raw](https://gist.github.com/lettergram/d2d329a6b6e9d7b87a6b/raw/d9fb8c0d58271aeba70d5cd020ff5150076408a0/mainwindow.h)[mainwindow.h](https://gist.github.com/lettergram/d2d329a6b6e9d7b87a6b#file-mainwindow-h)hosted with ❤ by [GitHub](https://github.com/)

mainwindow.cpp

|  |  |
| --- | --- |
|  | #include "mainwindow.h" |
|  | #include "ui\_mainwindow.h" |
|  |  |
|  | MainWindow::MainWindow(QWidget \*parent) : |
|  | QMainWindow(parent), |
|  | ui(new Ui::MainWindow){ |
|  |  |
|  | ui->setupUi(this); |
|  | ui->webView->load(QUrl("http://www.google.com")); |
|  | ui->webView->show(); |
|  |  |
|  | collect = false; |
|  | } |
|  |  |
|  | MainWindow::~MainWindow(){ |
|  | delete ui; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @brief MainWindow::on\_urlLineEdit\_returnPressed |
|  | \* - A URL has been inputed and will now load webpage |
|  | \*/ |
|  | void MainWindow::on\_urlLineEdit\_returnPressed(){ |
|  |  |
|  | QUrl url(ui->urlLineEdit->text()); |
|  | std::cout << "URL: " << ui->urlLineEdit->text().toStdString() << std::endl; |
|  | ui->webView->load(url); |
|  | ui->webView->show(); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* @brief MainWindow::on\_toggleButton\_released - Upon click |
|  | \* sets "collect" variable to true/false -- Start/Stop. |
|  | \* If collect is true, collect every object targeted/user input, |
|  | \* in order to produce selenium code. |
|  | \*/ |
|  | void MainWindow::on\_toggleButton\_released(){ |
|  |  |
|  | collect = !collect; |
|  | if(collect){ |
|  | ui->toggleButton->setText("Stop"); |
|  | }else{ |
|  | ui->toggleButton->setText("Start"); |
|  | } |
|  | } |
|  |  |
|  | // Back button for browser |
|  | void MainWindow::on\_backButton\_clicked(){ |
|  | ui->webView->back(); |
|  | } |
|  |  |
|  | // Generate Code upon release |
|  | void MainWindow::on\_genButton\_released(){ |
|  | // Nothing yet |
|  | } |
|  |  |
|  | void MainWindow::on\_webView\_selectionChanged(){ |
|  | if(ui->webView->page()->hasSelection()){ |
|  | // Log Focus |
|  | } |
|  | } |

  The code above currently does not have any of the code from goal #2, however it can function as a basic web browser, thereby accomplishing goal one. I can even view my website!

[](https://austingwalters.com/wp-content/uploads/2014/06/Screen-Shot-2014-06-08-at-10.16.25-PM.png)