Project report

Project Name: VSCE Extensions Creator

Project Manager's Name: Laurent Bouquin

Start Date: 2024-03-05

Last Updated: 2024-08-27

Status: Version 1.0.4

Introduction

This document is a report on the project. It includes:

- A comprehensive presentation of the candidate's professional project,
- The analysis and specifications of the software creation project,
- The choice of software architecture,
- An argument on the choice of algorithms and their relevance to the problems to be solved,
- A presentation of the tests developed by the candidate and the deployment of the software solution,
- The evolutions made or to be planned for the software solution,
- An analysis of the project management and management.

Project Overview

The project is a software solution that aims to solve a specific problem or address a particular need. The software solution is designed to be user-friendly, efficient, and scalable.

Problem Statement

After discussing with multiple developers, it was observed that when creating a new programming language, developers often face challenges in understanding the fundamentals of Visual Studio Code (VSCode) extensions. As VSCode extensions are an essential part of the development process, it is crucial to have a good understanding of how to create them. However, the process of creating VSCode extensions can be complex and time-consuming, especially for developers who are new to the platform. When you try to create a new programming language extension, you need to understand on which files you need to change in order to have the expected result.

Solution

This project aims to address this issue by providing an application that simplifies the process of creating VSCode extensions. The interface is divided into three main sections: the selector, the editor, and the settings.

Selector Section

The selector is where you can choose the extension you want to modify or create. The different extensions that the user created are displayed in the selector in a list format. The user can select the extension they want to modify or create a new one.

For each extension, the main information displayed are:

- The name of the extension (e.g. Python Language),
- The version (e.g. 2.3.7),
- The categories (e.g. Programming Languages),
- The description (e.g. A language extension for Python),
- And the last update date (e.g. 2024-08-15).

Editor Section

The second section is the editor, where you can customize the extension you selected / created. The editor is divided into five main sections: the format, the comments and strings, the snippets, the theming, and the miscellaneous.

Format Section

The format section is where you can configure the format of the extension. It includes options such as:

- File format,
- Indentation,
- Keywords,
- And more.

Comments and Strings Section

The comments and strings section is where you can configure the comments and strings of the extension. It includes options such as:

- Single line comments,
- Multi-line comments,
- Strings.

Snippets Section

The snippets section is where you can configure the snippets of the extension. It allows you to create custom snippets for the extension. You can add, edit, and delete snippets.

Theming Section

The theming section is where you can configure the theming of the extension. It lets you customize the colors for:

- Keywords,
- Strings,
- Comments,
- Background,
- Functions,
- And the base color.

Miscellaneous Section

The miscellaneous section is where you can configure other settings of the extension. It includes options such as:

- Auto-completion,
- Hover,
- Linting,
- Formatting,
- And more.

Settings Section

The settings section allows to configure the user preferences and the basic settings of a new extension. That part has been designed by thinking about users that would need to create multiple extensions that share the same basic parameters (e.g. the same file format, the same indentation, the same keywords, etc.). The user can save the settings and reuse them for other extensions.

Objectives

The main objectives of the project are to provide a user-friendly interface that simplifies the process of creating VSCode extensions, to improve the efficiency of developers in creating new programming language extensions, and to provide a scalable solution that can be easily extended to support new features and functionalities.

Side objectives include providing understandable user interface and user experience where the user can easily navigate through the different sections of the application, providing a clean and organized codebase that is easy to maintain and extend, and providing a secure and reliable solution that can be used in a production environment.

Target Audience

The target audience for the project is developers who are new to creating VSCode extensions or who want to improve their speed in creating new programming language extensions. The audience should include more junior developers who are looking to learn more about VSCode extensions, as well as students who are studying programming languages and want to create their own extensions.

Scope

The scope of the project includes the development of the software solution, the testing of the application, and the deployment of the solution. The project will be developed using modern technologies and best practices, and will be tested to ensure that it meets the requirements and objectives of the project. The deployment of the solution will be done in a secure and reliable manner, and will be monitored to ensure that it is working as expected.

Project Details

This part serves as a detailed analysis of the project, including the software requirements, the development environment, the technologies used, the conventions followed, and the technical choices made.

Development Environment

The software solution has been developed using a windows 11 environment and Visual Studio Code as the code editor. The application has been developed using Flutter and Dart at their latest versions on the date of the project. The application has been tested on multiple platforms including *Windows*, *MacOS* to ensure that it works as expected on different operating systems.

Technologies

The software solution has been developed using the following technologies:

- *Flutter*: Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase.
- *Dart*: Dart is a client-optimized programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, server, and web applications.

Conventions

The software solution follows conventions to ensure that the codebase is clean, organized, and easy to read.

Naming Conventions

The naming conventions used in the project are as follows:

- *CamelCase*: CamelCase is used for naming variables, functions, and classes. For example, `myVariable`, `myFunction`, `MyClass`.
- *PascalCase*: PascalCase is used for naming directories. For example, `MyDirectory`.
- *Pascal_Snake_Case*: Pascal_Snake_Case is used for naming documents files. For example, `My_Document`.
- *snake_case*: snake_case is used for naming files. For example, `my_file`.

Folder Structure

This folder structure only includes the main folders and files of the project. The actual project may have more files and folders depending on the requirements.

```
vsce_extensions_creator/
│
├── lib/
│   ├── customize/
│   │   ├── comments&strings.dart
│   │   ├── format.dart
│   │   ├── customisables.dart
│   │   ├── snippets.dart
│   │   ├── theme.dart
│   │   ├── miscellaneous.dart
│   │   └── functionalities.dart
│   │
│   ├── functional/
│   │   ├── functions.dart
│   │   └── convert.dart
│   │
│   ├── storage/
│   │   ├── commantsandstrings.json
│   │   ├── format.json
│   │   ├── snippets.json
│   │   ├── theming.json
│   │   ├── miscellaneous.json
│   │   ├── settings.json
│   │   ├── extensions_data.json
│   │   └── extensions_list.json
│   │
│   ├── main.dart
│   ├── home.dart
│   └── settings.dart
│
├── .gitignore
├── pubspec.yaml
└── README.md
```

Code

The code conventions used in the project are as follows:

- *Comments*: Comments are used to explain the purpose of the code, the logic behind it, and any other relevant information. Comments are written in English and are placed above the code they refer to.
- *Indentation*: Indentation is used to make the code more readable. The code is indented using one tab (4 spaces) for each level of indentation.
- *Spacing*: Spacing is used to separate different parts of the code. There is a space between keywords and parentheses, and between operators and operands.

Comments

Comments are used to explain the purpose of the code, the logic behind it, and any other relevant information. Comments are written in English and are placed above the code they refer to.

```
// This is a comment

// This function adds two numbers
int add(int a, int b) {

    // Return the sum of the two numbers
    return a + b;
}
```

Software Architecture

The software architecture of the project is designed to be modular, scalable, and maintainable. The architecture is divided into three main layers: the presentation layer, the business logic layer, and the data access layer.

Presentation Layer

The presentation layer is responsible for presenting the user interface to the user. It includes the user interface components such as buttons, text fields, and labels. The presentation layer is implemented using Flutter, which provides a rich set of widgets and tools for building user interfaces.

Business Logic Layer

The business logic layer is responsible for implementing the business logic of the application. It includes the logic for processing user input, performing calculations, and interacting with the data access layer. The business logic layer is implemented using Dart, which provides a powerful set of tools for implementing complex logic.

Data Access Layer

The data access layer is responsible for interacting with the data storage of the application. It includes the logic for reading and writing data to and from the data storage. The data access layer is implemented using JSON files, which provide a simple and efficient way to store data.

Software Effects

The software solution has a number of effects on the user, the developer, and the organization. These effects are both positive and negative, and should be taken into account when evaluating the success of the project. The effects of the software solution include:

- *User Effects*: The software solution provides a user-friendly interface that simplifies the process of creating VSCode extensions. Users can easily navigate through the different sections of the application and customize their extensions.
- *Developer Effects*: The software solution improves the efficiency of developers in creating new programming language extensions. Developers can save time and effort by using the application to create their extensions.
- *Organization Effects*: The software solution provides a scalable solution that can be easily extended to support new features and functionalities. The organization can use the application to create new extensions and improve their development process.

Error Handling

The software solution includes error handling mechanisms to ensure that errors are handled gracefully and that the user is informed of any issues that occur. Errors are handled using try-catch blocks, which catch exceptions and display error messages to the user. The error messages are written in English and are displayed in a user-friendly format.

Technical Constraints

The software solution has a number of technical constraints that must be taken into account when developing the application.

Security

The software solution does not store any sensitive information on the first version of the project. However, it is important to ensure that the application is secure and that user data is protected. The application uses JSON files to store data, these files are stored locally on the user's device and are not accessible to other users. The application does not send any data over the internet, so there is no risk of data interception.

Scalability

The software solution is designed to be scalable and to support new features and functionalities. The application is built using Flutter and Dart, which are modern technologies that provide a rich set of tools for building scalable applications. The application can be easily extended to support new features and functionalities, such as new sections, new settings, and new themes.

Maintainability

The software solution is designed to be maintainable and to allow for easy updates and modifications. The codebase is clean and organized, and follows best practices for software development. The code is well-documented and includes comments to explain the purpose of the code and the logic behind it. The code is written in English and follows naming conventions to make it easy to read and understand.

Metric Evaluation

The software solution is evaluated based on a number of metrics, including performance, reliability, feedback, and monitoring.

Performance

The performance of the software solution is evaluated based on the speed and efficiency of the application. The application should be responsive and should not lag or freeze when the user interacts with it. The performance of the application is monitored using performance testing tools and metrics.

Reliability

The reliability of the software solution is evaluated based on the stability and availability of the application. The application should be stable and should not crash or produce errors when the user interacts with it. The reliability of the application is monitored using error tracking tools and metrics.

Feedbacks

The feedback of the software solution is evaluated based on the user feedback and reviews. Users should be able to provide feedback on GitHub, and the feedback should be used to improve the application. The feedback of the application is monitored by the development team and used to make updates and modifications to the application.

Monitoring

The monitoring of the software solution is evaluated based on the monitoring tools and metrics used to track the performance and reliability of the application. The monitoring tools should provide real-time data on the performance and reliability of the application, and should be used to identify and resolve any issues that occur.

Algorithms and Problem Solving

The software solution includes algorithms to solve specific problems and to improve the efficiency of the application. The algorithms are designed to be fast, efficient, and scalable, and to provide accurate results.

Algorithms

The software solution includes algorithms mainly to generate the final files that will be used to create the extension. The algorithms are designed to process the user input and generate the output files in a timely manner. The algorithms are implemented using Dart, which provides a powerful set of tools for implementing complex logic.

Problem Solving

The software solution includes problem-solving techniques to address specific issues and to improve the efficiency of the application. The problem-solving techniques are used to identify and resolve issues that occur during the development process, and to improve the performance and reliability of the application.

Testing and Deployment

Unfortunately, the first version of the project does not include any testing. However, the project has been tested on multiple platforms including *Windows*, *MacOS* to ensure that it works as expected on different operating systems. The deployment of the solution will be done in a secure and reliable manner, and will be monitored to ensure that it is working as expected.

Testing

Starting from the second version of the project, the software solution will include testing to ensure that it meets the requirements and objectives of the project. The testing will include unit testing, integration testing, and end-to-end testing to ensure that the application is working as expected. The testing will be done using testing tools and frameworks to automate the testing process and to provide accurate results.

Deployment

The deployment of the solution will be done in a secure and reliable manner, and will be monitored to ensure that it is working as expected. The software will be available on GitHub and be downloadable for *Windows* and *MacOs* . The deployment process will include testing the application on different devices and platforms to ensure that it works as expected.

Evolutions

The software solution will be updated and improved over time to add new features and functionalities, to fix bugs and issues, and to improve the performance and reliability of the application. The evolutions of the software solution will be done in a timely manner and will be based on user feedback and reviews.

Current State

The current state of the software solution is the first version of the project. The software solution includes the main features and functionalities of the application, and is designed to be user-friendly, efficient, and scalable. The software solution is available on GitHub and is downloadable for *Windows* and *MacOs*.

Future State

The future state of the software solution will include new features and functionalities, such as new sections, new settings, and new themes. The software solution will be updated and improved over time to add new features and functionalities, to fix bugs and issues, and to improve the performance and reliability of the application. The future state of the software solution will be based on user feedback and reviews, and will be designed to meet the requirements and objectives of the project. The future versions of the project will also include snippets and themes that can be fully customized by the user.

Project Management

The project management of the software solution includes planning, monitoring, communication, risks, and lessons learned.

Planning

The planning of the project is separated into different phases:

- *_Phase 1: Requirements and Analysis*:
    - *Requirements*: The requirements phase includes the identification of the main requirements and objectives of the project, and the definition of the scope and constraints of the project.
    - *Analysis*: The analysis phase includes the analysis of the requirements and objectives of the project, and the identification of the main features and functionalities of the application.
    - *Design*: The design phase includes the design of the software architecture, the user interface, and the algorithms of the application.
    - *Develop V1's features*: The development phase includes the development of some of the main features and functionalities of the application, and the testing of the application to ensure that it meets the requirements and objectives of the project.
    - *Test the V1*: The testing phase includes the testing of the application to ensure that it meets the requirements and objectives of the project. It also includes the crash tests and the performance tests.
    - *Deployment of the V1*: The deployment phase includes the deployment of the solution in a secure and reliable manner, and the monitoring of the solution to ensure that it is working as expected.
- *_Phase 2: Evolutions and Improvements*:
    - *Analysis*: The analysis phase includes the analysis of the new requirements and objectives of the project, and the identification of the new features and functionalities of the application.
    - *Develop V2's features*: The development phase includes the development of the new features and functionalities of the application, and the testing of the application to ensure that it meets the new requirements and objectives of the project.
    - *Test the V2*: The testing phase includes the testing of the application to ensure that it meets the new requirements and objectives of the project. It also includes the crash tests and the performance tests.
    - *Deployment of the V2*: The deployment phase includes the deployment of the solution in a secure and reliable manner, and the monitoring of the solution to ensure that it is working as expected.

Monitoring

The monitoring of the project includes tracking the progress of the project, identifying any issues that occur, and resolving them in a timely manner. The monitoring of the project is done using project management tools and metrics to ensure that the project is on track and meeting the requirements and objectives of the project.

Communication

The communication of the project includes regular updates and reports on the progress of the project, and the sharing of information and feedback with the development team. The communication of the project is done using project management tools and channels to ensure that the project is on track and meeting the requirements and objectives of the project. In the case of the development team finding companies that would be interested in the project, the communication will be done with the companies to present the project and discuss the potential collaboration.

Risks

The risks of the project include technical risks, and organizational risks. The risks of the project are identified and analyzed to ensure that they are mitigated and managed in a timely manner. The risks of the project are monitored using risk management tools and metrics to ensure that the project is on track and meeting the requirements and objectives of the project.

Lessons Learned

The lessons learned from the project include the main challenges and issues that occurred during the development process, and the main successes and achievements of the project. The lessons learned are used to improve the development process and to ensure that the project is on track and meeting the requirements and objectives of the project.

Conclusion

The software solution is designed to be user-friendly, efficient, and scalable. The application provides a user-friendly interface that simplifies the process of creating VSCode extensions, improves the efficiency of developers in creating new programming language extensions, and provides a scalable solution that can be easily extended to support new features and functionalities. The software solution is designed to meet the requirements and objectives of the project, and to provide a secure and reliable solution that can be used in a production environment.

References

- Flutter
- Dart
- Visual Studio Code
- GitHub
- Windows
- MacOS
- Google