



College of Engineering

FALL TERM CS CAPSTONE DESIGN DOCUMENT

FEBRUARY 4, 2020

SOFTWARE APPLICATION FOR ASUS DUAL SCREEN NOTEBOOK

PREPARED FOR

INTEL

MIKE PREMI

PREPARED BY

GROUP 66A

DUO TECH

SACHIN SAKTHIVEL

ROSALINDA GARCIA

DERK KIEFT

MATTHEW FERCHLAND

Abstract

This document describes the design components of the Intel sponsored Asus ZenBook Pro Duo application. It will contain an overview section discussing the purpose, audience, definitions and the context of the project. This is followed by the core design principles that will be used to develop the application, which includes the design stakeholders, viewpoints and rationale. The final section will describe the four major components of the application: window handlers, saving configurations, retrieving configurations and GUI framework. This section will be explained in depth and related back to the design description.

CONTENTS

1	Overview	3
1.1	Scope	3
1.2	Purpose	3
1.3	Intended Audience	3
2	Definitions	3
3	Project Context	3
3.1	Hardware	3
3.2	Software	4
4	Design Description	4
4.1	Design stakeholders	4
4.1.1	Intel Sponsor (Mike Premi)	4
4.2	Design views	4
4.2.1	Users	4
4.2.2	Intel Sponsor (Mike Premi)	4
4.3	Design viewpoints	4
4.3.1	Context viewpoint	4
4.3.2	Composition viewpoint	5
4.3.3	Dependency viewpoint	6
4.3.4	Interface viewpoint	6
4.4	Design Rationale	7
5	Approach	7
5.1	Gathering and Manipulating Window Handler Information	7
5.1.1	Concerns	7
5.1.2	Approach	7
5.2	Improve Productivity by Saving Configurations	8
5.2.1	Concerns	8
5.2.2	Approach	8
5.3	Improve Productivity by Retrieving Saved Configurations	8
5.3.1	Concerns	8
5.3.2	Approach	9
5.4	Efficient and Visually pleasing GUI Interface	9
5.4.1	Concerns	9
5.4.2	Approach	9
6	Timeline	10

LIST OF FIGURES

1	Dependency Graph of the Shell Configuration App.	6
2	Tentative Gantt Chart for the year.	10

1 OVERVIEW

1.1 Scope

The goal of this project is to create an overarching shell program that will allow users to save, retrieve and share their custom window handler configuration. This shell will then be used in conjunction with other applications to optimize productivity within the companion screen of the Asus ZenBook Pro Duo. In addition, the shell program will also have resizing features specifically designed for the second screen on the notebook.

1.2 Purpose

Due to the limited applications that are compatible with the companion screen of the laptop, it was proposed that the main purpose of this project is to build an application that allows other programs to properly accommodate the second screen. This document exists both for development of the project and to provide a detailed overview of the design plans.

1.3 Intended Audience

The intended audience of this document are the student developers that are working on this project (Tech Duo), stakeholders, sponsors and the CS 461 professors. The developers of this project can use this document as a guide to help formalize and structure the code implementation. Stakeholders and sponsors will read this document to understand the goals of the developers and how it can be achieved. Finally, the professors will use this document for grading and to validate that the design plan is acceptable.

2 DEFINITIONS

- **Companion Screen:** 4K 32:9 IPS "ScreenPad Plus" screen directly above the keyboard
- **Main Screen:** 4K 15-inch 16:9 OLED touch display screen on the face of the notebook
- **C#:** A programming language developed by Microsoft that specializes in UI development
- **Notebook:** The Asus ZenBook Pro Duo laptop
- **OS:** Operating System
- **GPU:** Graphics Processing Unit
- **CPU:** Central Processing Unit
- **GUI:** Graphical User Interface
- **Configuration:** Layout (size, position) of different windows
- **Window Handler:** size, location, unique ID and name of an application's window

3 PROJECT CONTEXT

3.1 Hardware

The hardware that will be used for the final application is the Asus ZenBook Pro Duo which has the following specifications:

- **Main Screen:** 15-inch 16:9 OLED panel
- **Companion Screen:** 32:9 IPS "ScreenPad Plus"
- **CPU:** Intel Core i9-9980HK

- **GPU:** NVIDIA GeForce RTX 2060
- **Memory:** 16/32GB 2666Mhz DDR4
- **OS:** Windows 10

For testing, the developers will use a separate dual screen monitor with touch screen capabilities.

3.2 Software

- **Visual Studio:** Used to develop C# code and also has the XAML Designer program that will be used to create the GUI
- **GitHub:** Used by developers to collaborate, save, retrieve and share files

4 DESIGN DESCRIPTION

The following section will discuss the stakeholders, views, viewpoints and rationale behind the design of the project.

4.1 Design stakeholders

4.1.1 Intel Sponsor (Mike Premi)

In this project, Intel sponsor, Mike Premi, is working with nine students to help create applications that will improve the productivity of the ZenBook laptop. Mike will also provide all necessary hardware to test and validate our applications.

4.2 Design views

4.2.1 Users

Users of this product expect that the shell window configuration application will help automate work space environments and improve overall productivity. The shell program will be compatible with all types of window handlers and monitor sizes. In addition, it is also crucial that the main application has ZenBook specific features because this will be the primary user platform. Users will also expect that the GUI interface is visually appealing and easy to manage. In other words, users should be able to quickly learn and master the program after a few uses. Experienced users will also want more advanced options that provide more freedom in their custom configuration modes. Finally, users will expect that the shell program is not CPU intensive and does not conflict with any other application.

4.2.2 Intel Sponsor (Mike Premi)

The sponsor of this project, Mike Premi, is mainly concerned with the technical side of the application. For example, Mike would like to understand the language for development, GUI tool for usability, necessary hardware, performance metrics, target audience and overall technical design decisions. That being said, this document should discuss all of these components and allow Mike to verify if it meets his expectations.

4.3 Design viewpoints

4.3.1 Context viewpoint

The context viewpoint involves the UI/UX of the software. In this context, the software can be considered from a users point of view without thinking about the programming behind the UI.

Design Concern: The primary design concern involves the UI for the shell program. Pre-existing shell programs that perform similar functions provide confusing user interfaces. With the number of settings that can be included in the program, a concern is that users will struggle to navigate the numerous buttons and menus to perform simple actions. Additionally, a lesser concern is that users will prefer other software based on visual appeal.

Analytical Methods: Based on the above concerns, this UI should provide an easy to use and easy to understand experience for the user. This means that processes should be intuitive (i.e. buttons and menus should be labelled clearly). The development will be completed with this goal in mind. Then, informal user testing will help us verify that the program is intuitive and well designed. A variety of users of different backgrounds (e.g. various genders, various technical backgrounds) will be included in this process in order to rigorously test the UI. Multiple user tests may be run if the feedback indicated a need for large scale changes or updates to the design. Additionally, once the UI is designed and finalized, an instruction set or tutorial will be provided in the app for users who may want to view it before beginning to use the application and to also address user's problems and questions.

Rationale: An important feature of this shell program is that it is user friendly. As previously mentioned, many related products for window and monitor management use basic UIs that are crowded and confusing. The features are not pleasing for users and the number of options presented can confuse a user who is unfamiliar with the application. Thus, a benefit to using this software over others is a better experience and lesser learning curve. Additionally, allowing users to be able to find all of the necessary information and answers in the app without needing to look up information on the internet will increase the ease of use of the product.

4.3.2 *Composition viewpoint*

This viewpoint considers the interactions between different technologies and components within the product. As this product is entirely software, the different components are the program files of the software itself, the Windows operating system, and the Asus ZenBook Pro Duo laptop.

Design Concern: The interaction of these components will determine the performance of the software. A concern for this viewpoint is that the interactions between components will not be smooth and may result in unexpected behavior. It will be difficult to ensure that the components interact correctly as there is no existing framework development on the ZenBook Pro Duo. If these components do not interact properly, there may be unexpected behavior (i.e. application windows moving to the wrong location or settings being changed unnecessarily) and delays in functionality.

Analytical Methods: During development, these interactions will be programmed to only take place when necessary. The number of interactions will be limited by utilizing the Win32 API library. These interactions will be tested using manual testing and automated testing. This will allow the team to observe and record any unusual behavior in order to find the cause and create a fix. However, unintentional behaviors may only be observable in specific contexts, so it will be difficult to ensure that the interactions of all of the components is proper in all cases. As mentioned previously, user testing will take place to improve the user interface. However, this will also aide in finding bugs in the interactions as users tend to find bugs that developers may overlook or not encounter.

Rationale: This context is required for proper and efficient functionality which is essential to the success of the project. Additionally, understanding these interactions will lead to faster debugging and more efficient code. This

will benefit the developers and the stakeholders as development tasks will be accomplished faster.

4.3.3 Dependency viewpoint

The Dependency viewpoint defines the major dependencies and relationships among the entities within the application. The relationships will include shared data, compartmentalization of functionality and order of operations which will determine key elements of workflow.

Design Concern: The primary design concern of the dependency viewpoint is the flow of tasks to be completed in any workflow. Clearly stated dependencies will reduce the occurrence of redundancies while saving time and resources during development. The modules and aspects with the least dependencies will be developed later in the project to ensure the maximum functionality is achieved in the course of the project.

Analytical Methods: Clear workflow models will help determine initial dependency requirements which will be modified throughout the development process in order to refine components. During testing phases, we will be able to finalize any further dependency modifications required.

Rationale: The dependency viewpoint will be an important element of this project because we intend to create a robust and versatile system which can reliably interact with any software in multiple configurations. Ensuring that the dependencies are well designed will lead to stable and reliable software. Much of the functionality in the project will revolve around the ability to save and load configuration properties. This requires that the Graphical User Interface acts as the main hub for creating a saved configuration file and loading a previously saved configuration.

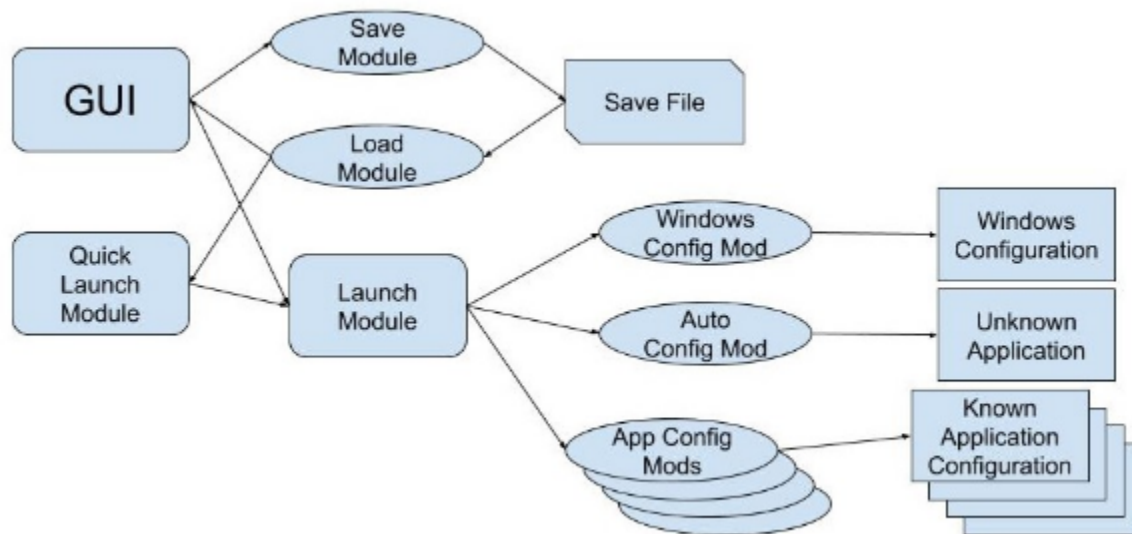


Figure 1: Dependency Graph of the Shell Configuration App.

4.3.4 Interface viewpoint

The main Interface for this project will include a configuration setup window where layout settings can be established. Once a layout is configured, there will be a “one-click” startup method that will restart a previous configuration without opening the main configuration interface. The purpose of this viewpoint is to clarify the interactions of the user interface.

Design Concern: The main concern regarding this viewpoint will be ease of use in configuration of layouts and the ability to save and reopen them. Further interface features will be from the programs that run within the shell and will not be altered by the shell itself. Because the interface will be the key to allow any functionality to the shell application, it is imperative that it is scalable. In addition, because of the potential scope of features added throughout the project and in the future, the interface will need to be designed to maximize the use of screen space.

Analytical Methods: Initially, the focus will be on creating a functional Graphical User Interface that allows the configuration of multiple layouts across the main and companion screens. Once the initial interface is functional, we will begin expanding into more advanced options to maximize configuration options and usefulness. Because this will be an iterative process, we will be utilizing an Agile development framework.

Rationale: This viewpoint is key to this project's ability to deliver a useful shell that is configurable to work with many applications and is expandable for future development. It is also important to establish the ability for users to quickly interact with various configurations.

4.4 Design Rationale

We chose to develop our application using C# in Visual Studio. We have listed several reasons why we chose these options below.

- C# and Visual Studio are well integrated by Microsoft.
- C# has an API with the ability to control window handlers.
- Windows, C#, and Visual Studio are all Microsoft products, which allows a closer abstraction layer for development.
- Several of our team members have experience with C#.
- In the project description, the client requested that we use C#.

5 APPROACH

5.1 Gathering and Manipulating Window Handler Information

The first step to developing the shell configuration application is understanding the Win32 library and how C# interacts with different applications. This includes getting a window's size, location and name.

5.1.1 Concerns

A primary concern that comes with window handler information is whether we will have the freedom to manipulate and store a window's information. For security reasons involving the operating system, it may be difficult to achieve certain functionalities. That being said, certain components that interact too closely with the kernel may not be possible.

5.1.2 Approach

The most integral components that we need to understand in order to successfully develop our program are listed below.

- We first need to learn how to create a basic Windows 10 application that can gather window handler information using the GetWindowInfo function from the Win32 library.

- The next step is to understand how we can use the stored window handler information to move and resize a window.
- After we figure out how to move a window and change its size, we will need to figure out how to control the windows on multiple screens at once. For example, allowing the user to move various windows to the companion screen.

5.2 Improve Productivity by Saving Configurations

It is imperative for the shell configuration application to have the ability to save a custom user configuration. Users need to have the ability to save the location and size of various windows to be able to load the configuration at a future time.

5.2.1 Concerns

The primary concern with saving a configuration is how the data will be managed. It doesn't seem necessary to have an external database to store various configurations because the amount of data that needs to be saved is very minimal. However, as the product becomes more popular, having an external database can be useful to ensure user data isn't lost.

5.2.2 Approach

There are a few possible ways to handle the approach of saving a configuration. We have listed some of them below:

- Based on the concern discussed above, the primary way to save a configuration will be to use a local file system. This would avoid the overhead cost of using a large-scale database management system.
- We will also want a general user preference file, or several, that the user can use to specify custom window handler settings. When creating a configuration session, the user would be able to specify which preference file they want to use, which would control things like focusing certain windows.
- The final step for this approach is to allow users to share a configuration session with another person. This feature will provide the recipient a local copy of the sender's configuration. The recipient would then be able to make changes to that copy based on personal preferences.

5.3 Improve Productivity by Retrieving Saved Configurations

Any saved configuration setting (see section 5.2 Improve Productivity by Saving Configurations) will need to be reloaded by users using a single click method without opening the main configuration page. Layouts will also need to be loadable in the main configuration interface in order to allow duplication and modification.

5.3.1 Concerns

The users of the application will not want to configure a new session each time the application is opened and will also not necessarily want to continue with the last layout used. Therefore, we will need to develop a method to save configurations and subsequently recreate the layout later without the need to revisit the configuration screen. The layout configuration will also need to be loadable into the configuration screen (without loading the layout) in order to allow it to be modified by the user.

5.3.2 Approach

The approach used to restore saved layouts will entirely depend on the method of saving the information. The application will need to step through the layout configuration and “load” in each setting into the shell as it executes each required application.

- This element will be developed alongside the layout configuration save functionality to ensure that user settings are saved properly and is recoverable when reopened.
- It will also be required to develop a logical order of operations for reloading a saved layout. For example, a layout can be changed only after it is reloaded.
- We will need to develop a system to ensure that all previous dependencies are met before certain elements attempt to load.
- The Win32 Library will be required to configure window positions using `SetWindowPos` to resize and move a window handler to specific locations.
- Virtual Desktop functionality will also be explored to determine if its tools will be useful in setting up pre-configured application sets.

5.4 Efficient and Visually pleasing GUI Interface

As previously discussed in section 4.3.1, a part of this development process will include creating a good user experience for the software. This includes intuitive designs such as well labelled buttons and well organized menus. This also includes creating a customizable experience for users with options and settings as well as shortcuts as part of the Graphical User Interface (GUI).

5.4.1 Concerns

The GUI will need to support multiple users of different experience levels. Thus, a concern with the development of the GUI is that it will work for only certain users and will result in some users struggling to perform the actions they want. The product should allow for a customizable experience so that any type of user can make a configuration that is helpful for them. Considering that there is only a short time for developing and testing, there is a likelihood that a subset of the intended audience may struggle with using the UI. This leads to a concern that even if there is a plan for accommodating multiple users, there will not be time to make the desired fixes.

5.4.2 Approach

Based on these concerns, the UI for the software will be prototyped first. The team will discuss the prototype and make adjustments with various users in mind. This process will be completed primarily using XAML Designer for Visual Studio. The following are some components of the UI/GUI.

- A variety of options will be available to allow the user to create a customized experience and to create a larger amount of functionality. These options will include both simple options for more common use cases as well as advanced options for a more in-depth functionality. Using these options, the user will be able to change the default functionality for when the program opens/starts. This will allow users to customize the UI and program to their needs.
- Users will be able to customize their shortcuts and hotkeys for retrieving previous settings and performing simple actions such as applying a previously saved layout to a display or moving a window to the companion

screen. The purpose of the software is to increase the usefulness of the companion screen, so the user should be able to make use of this display without going through multiple layers of menus to find the correct settings. This can be taken one step further by allowing users to customize their own hotkeys and shortcuts for easy use. The GUI for this will be simple and straightforward with the minimal number of buttons/selections.

- The design will be sleek and easy to use. Many products with similar functionality have UI designs that are not visually pleasing. Instead, this design will involve better organization as well as a modern GUI that is appealing. This will make the software easier and more enjoyable to use, contributing to a better user experience.

6 TIMELINE

This section includes a tentative Gantt chart that discusses the current progress plan for the application. If the overall shell application is built before expected, the developers will create additional programs using the shell.

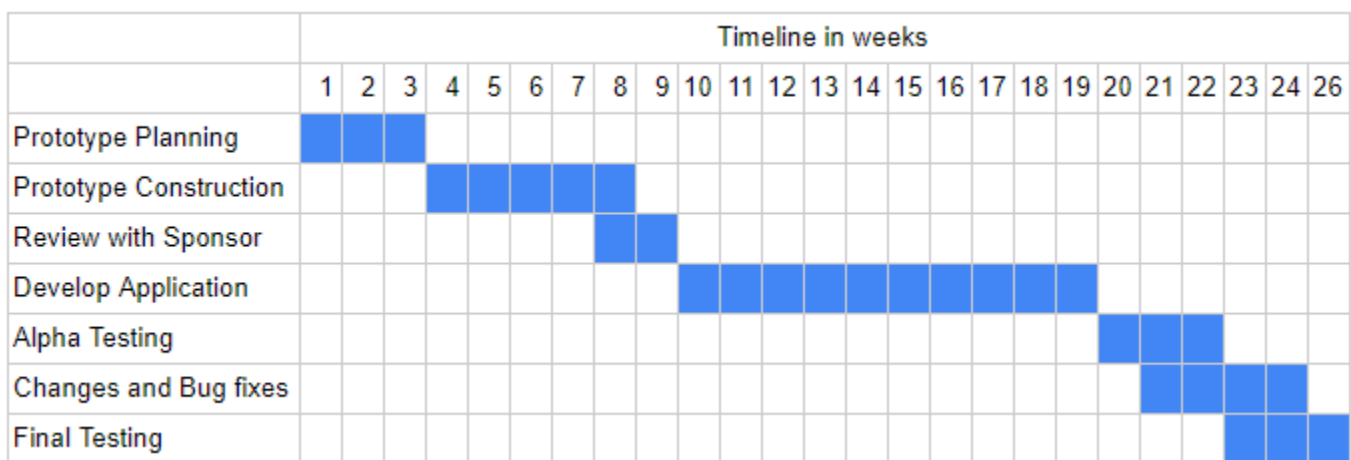


Figure 2: Tentative Gantt Chart for the year.