



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

DEPARTMENT OF
COMPUTER SCIENCE
Te Tari Rorohiko



2024

Contributors

J. Turner

R. Mercado

V. Moxham-Bettridge

© 2024 University of Waikato. All rights reserved. No part of this book may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without prior consent of the Department of Computer Science, University of Waikato.

The course material may be used only for the University's educational purposes. It includes extracts of copyright works copied under copyright licences. You may not copy or distribute any part of this material to any other person, and may print from it only for your own use. You may not make a further copy for any other purpose. Failure to comply with the terms of this warning may expose you to legal action for copyright infringement and/or disciplinary action by the University.

THE SOFTWARE DEVELOPMENT LIFECYCLE: IMPLEMENTING THE UI

Last week we explored human-robot interaction with our robot, Cruz. This week we move onto the initial implementation phase of the SDLC. In particular, we will look at building the user interface using Visual Studio.

From Prototyping to Software

Once a prototype of the software has been created, the next step is to actually implement it! We take our prototypes and convert these into actual user interfaces for the software we are developing.

Consider the screenshot from a model-checking software.

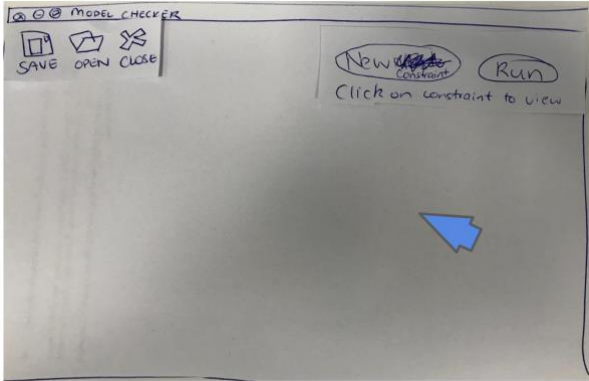
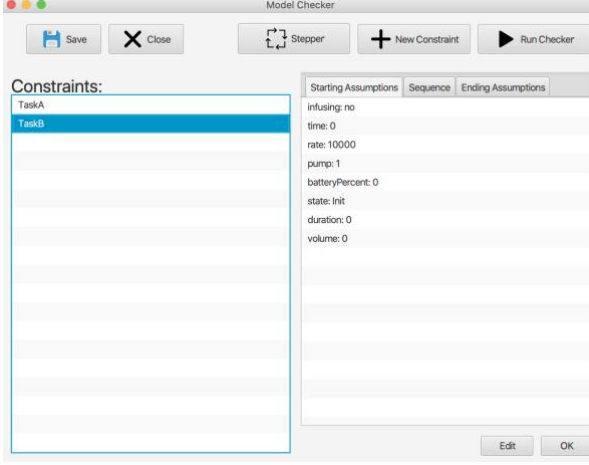
	
Paper Prototype	Implemented Software

Table 2: Software Model Check Implementation

Notice that the implemented software is far more detailed than the paper prototype. There are more areas of the user interface that are filled in and more options available.

Because of the messiness of paper prototyping the software that we implement may not necessarily look exactly the same as the prototype. Furthermore, as we create the user interface new ideas or issues may emerge that need to be addressed. This is perfectly normal as part of the

development process. Software is not fixed, it should be continually updated and improved in order to provide the best possible user experience.

In a typical development environment, teams of people would develop the software for the client. These teams can have many different names but a common distinction is a front-end or back-end developer.

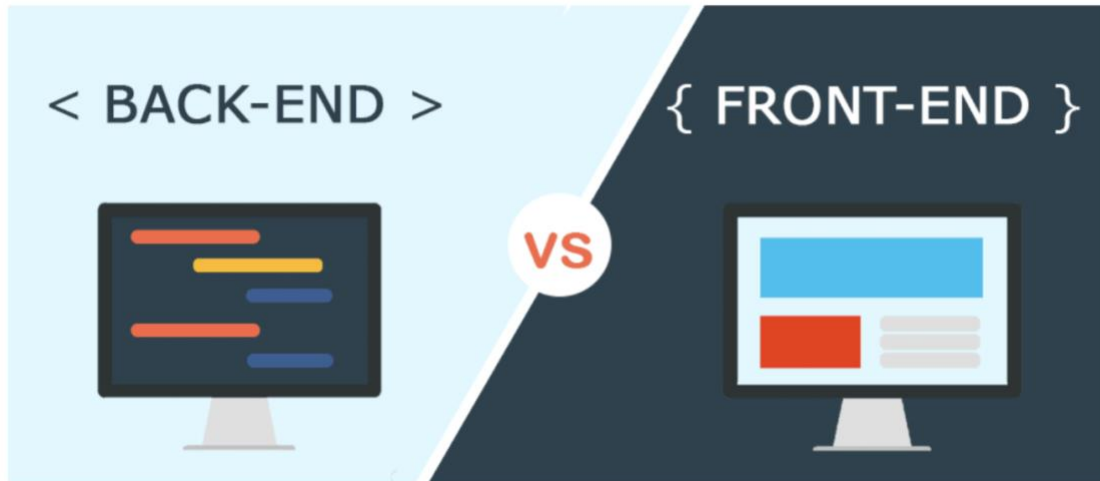


Figure 10: Back-end vs. Front-end Development. Retrieved from <https://dev.to/molly/frontend-vs-backend-which-one-is-right-for-you-5gjg>

The front-end developer usually works on the visual elements of an application, the part that the user will interact with. The back-end developer does the opposite and works on the side that the user can't see. For example, in a website application, the front-end developer might work with HTML, CSS and Javascript to get the application to look and respond to user's actions in a particular way. On the other hand, the back-end developer might work with Javascript, PHP, SQL, and so on in order to ensure the functionality behind the application works as intended.

By the time you have completed all of the CSNeT Beyond Beta sessions you will have experienced both front and back-end development, albeit in a simplified setting. In today's session you are going to create the front-end of the software we are building for Parkway Commute. To do this you will use the paper prototype provided on the Slack channel to guide you through the front-end development.

C# User Interfaces

Today you will be working on a Windows Forms application using the .NET framework. It is important that when the session is over that you save the project folder in a safe place (e.g. Google Drive) so that you can access it in the next session.

Exercises:

1. Download the project from the Slack channel and open it in Visual Studio.

When the project opens you will have one form by default. For today's session you will need to create more than one form, the number of forms you choose to implement depends on how far you progress with the prototype.

2. Have a go at creating a new form for your project. Right click on the project name and go to the "Add" submenu. From the add submenu select "Form (Windows Forms) ...". Follow the prompts and then click add. If completed successfully your project explorer will show the following:

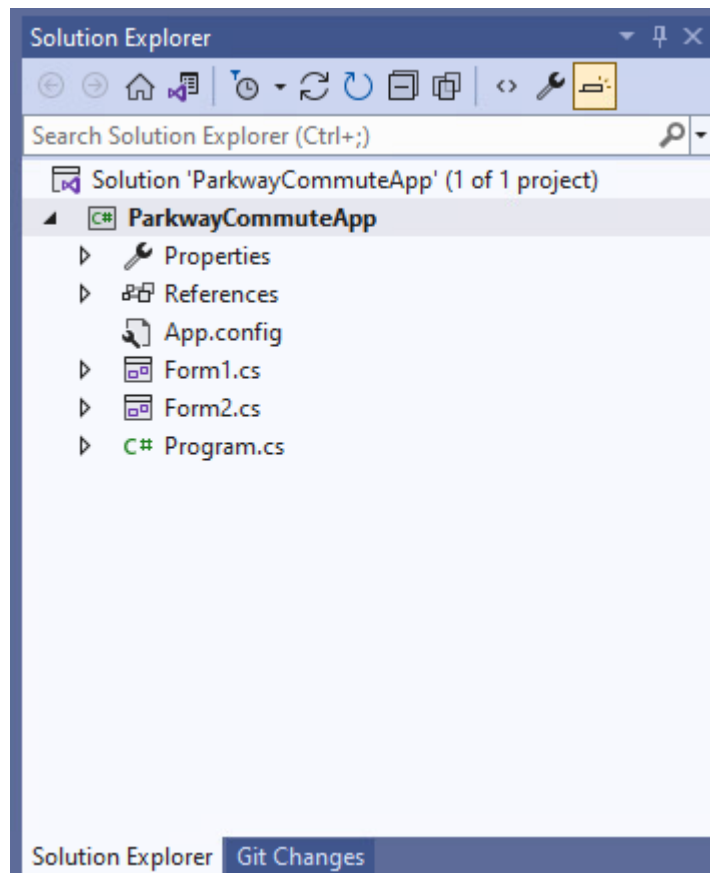


Figure 12: ParkwayCommuteApp with two forms

Next, let's inspect the toolbox. The toolbox allows you to add different widgets to the screen. If you select the "All Windows Forms" option you will see all possible widgets. Some of the most commonly used widgets include buttons, check boxes, labels, and text boxes. You can add widgets to your form by clicking and dragging them onto the design space.

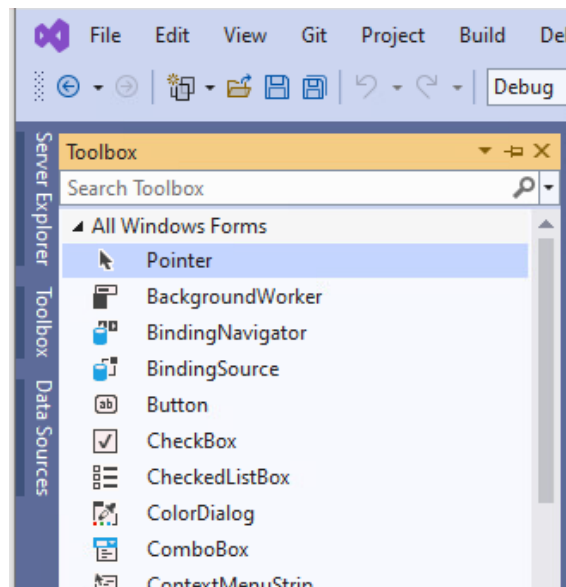


Figure 13: The Toolbox

3. Try dragging and dropping a button onto one of your forms.

Note in the properties window that your button should be called "button1". It is best practice to give widgets more descriptive names. You can change the text displayed on the button by changing the button's "Text" property. You can change the button's name by changing the button's design name.

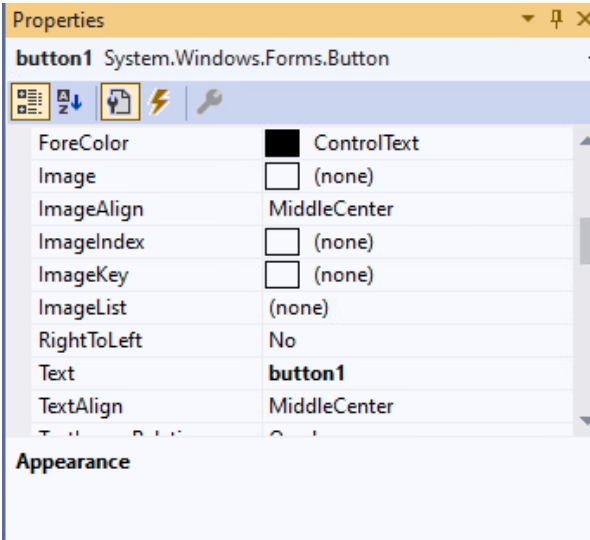


Figure 14: Text Property

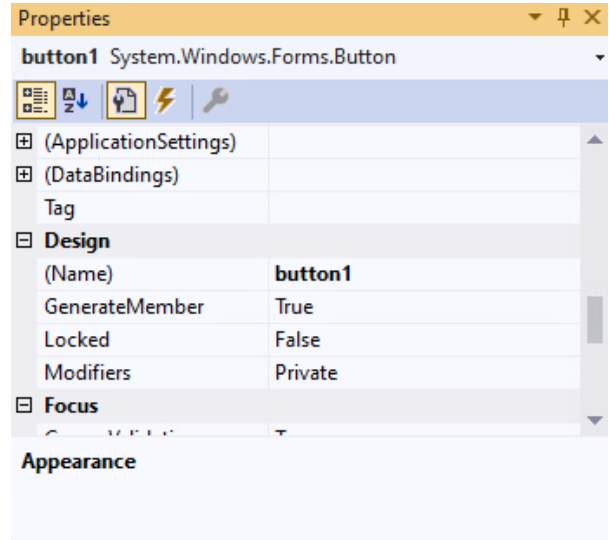


Figure 15: Name Property

If you click on the event's button in the properties window (the lightning bolt button) you can specify different actions for the button such as Click, MouseDown and so on. With our two forms we are going to demonstrate the code for switching between two windows.

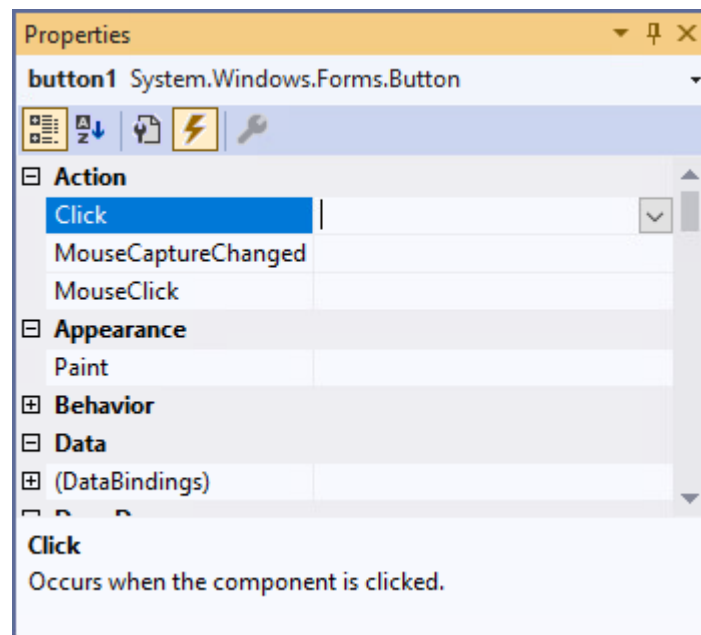


Figure 16: Events

4. Add a button to form one. The properties of the button should be as below. Repeat the same for form two.

Text: Display Form Two



Name: buttonDisplayFormTwo

5. Locate the Form1 Design and double click on the button. This will automatically add a click event for the display button. Complete the code with the following:

```
private void buttonDisplayFormTwo_Click(object sender, EventArgs e) {  
    this.Hide();  
    Form2 f2 = new Form2();  
    f2.ShowDialog();  
}
```

The above code will hide the current form, create a new form and then tell the computer to display the new form. Repeat the same for form one and ensure that you can transition between the two different forms.

You should notice that as you transition between the forms they appear in different places on the screen. There are two ways to get around this. The first is to use the full screen, the second is to specify where you want the form displayed.

6. Modify your Form 1 code as below. Ensure you do the same for the Form 2 code.

```
private void buttonDisplayFormTwo_Click(object sender, EventArgs e) {  
    this.Hide();  
    Form2 f2 = new Form2();  
    f2.StartPosition = FormStartPosition.CenterScreen;  
    f2.ShowDialog();  
}
```

Now you have the necessary skills to build your user interface in C# by selecting the appropriate widgets and moving between forms as above.

Today's Exercise

Today's exercise is to use the skills you have learned above to implement the user interface of the prototype for the Parkway Commute project (the paper prototype is provided on the Slack channel). If you aren't sure which widgets to use or get stuck then ask a staff member for assistance. At the end of the session remember to save your project folders in a safe place. Ask a staff member which files you need to save.



Summary

Today we started working through the implementation process of the SDLC based on the design artefacts that you created in earlier sessions. Next session we will continue with the implementation phase and start building up some of the functionality of your application.

Useful Resources

- Create a Windows Forms app in Visual Studio with C#: <https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022>
- W3Schools C# Tutorial: <https://www.w3schools.com/cs/index.php>
- C# Windows Form App with Class: <https://www.youtube.com/watch?v=774h956uIQs>

