|  |  |
| --- | --- |
| Assessment Title | Assignment (Universal Calculator) |

## Competency Details

|  |  |
| --- | --- |
| Unit code/s and title/s | ICTICT449 Use version control systems in development environment |
| Qualification code/s and title/s | ICT40120 Certificate IV in Information Technology |
| Business unit/Work team | Business and Arts/IT Studies |

## Instructions

|  |  |
| --- | --- |
| Method/s of assessment | Product (Written) |
| Overview of assessment | This assessment will require you to complete four sections requirements. Each section has multiple parts require you to complete.  In this assessment you will cover the following topics:   * Prepare to use the version control system * Build files in version control * Test the version control system * Finalise the use of the version control system   The assessment is broken into 6 separate SECTIONS (SECTION A ~ F) and you will submit the released product by the assessment due date. There are some questions are team works and some questions are individual works. Students must make sure which question requires you to participate as team work and/or individual work.  You will also have a meeting with your IT Manager (your lecturer) to discuss the planning and the final product to ensure you understand the industry practices. |
| Task/s to be assessed | This assessment will require you to complete the following tasks   * SECTION A. (Team Work - oral and written VCS requirements) * SECTION B. (Team Work – Configure the VCS) * SECTION C. (Individual work) - Develop features using branching and merging * SECTION D. (Team Work) – Preparing for product release * SECTION E. (Individual Work) – Document the test outcomes * SECTION F. (Individual Work) – Create an installation disruption notification |
| Time allowed | Refer to your schedule for submission dates |
| Location of assessment | Assessment can be completed anywhere with access to the resources required. (see Resources Required section below) |
| Decision making rules | To receive a satisfactory outcome for this assessment you must complete all parts correctly.  Word counts are provided as guidance only. |
| Assessment conditions | This assessment must be undertaken where the conditions replicate noise levels and interruptions that people typically experience working in the ICT industry.  This is unsupervised assessment, and you may access any required resources.  There are group work component and individual component. Group work must be completed with team members as a group. Individual work must be completed as an individual. |
| Resources required | To complete this assignment, you will need to use Git for Windows. Git for Windows, TortoiseGit, Visual Studio and Windows based machines are provided in your practical classes. You can use a Mac if you prefer but these are not provided in the classrooms.  You will also need to contact your IT Team Leader either in class or have access to collaborate. |
| Result notification and reassessment information | You will be provided feedback and the result for your assignment on TAFESA Learn. You will be and given the chance to resubmit with required corrections only once.  Refer to the TAFE SA assessment policy for more information <https://www.tafesa.edu.au/apply-enrol/before-starting/student-policies/assessment> |

**Assignment (Universal Calculator)**

***Assignment background:***

You have been employed by the ITWorks organization as a junior programmer working under the guidance of Joe Black who is the IT Software Development Manager (hereinafter referred to as The Manager). Your job is to further develop an open-source Calculator application so that the employees would be able to use it without cost.

The original application is not a complete product as it has only a limited set of Calculator functions. The project was hosted under the GitHub open-source community server. Now the Calculator application has been exported and free from any version control. It is now ready for you to put it under the administration of the ITWorks version control repository for further development. The project will continue to be an open-source project hosted within a public repository. You will team up with other junior programmers to further develop the application.

ITWorks has several projects which are hosted in the GitHub for the open-source community to access and share. The Git version control software was chosen because it is a distributed version control system which is more suited for multiple developers develop application at the same time.

ITWorks has installed a similar calculator application (i.e. The calculator V1.0 that you have developed in the previous assignment) on the staff machines. However, the application is missing other features and will be uninstalled and re-installed with the new Universal Calculator application with more features.

The given Universal Calculator application has been completed with the development on one functional area such as maths calculator. The project will be further developed by your team members. Each team member must choose to develop one of the functional areas from the following:

The team also needs to modify the existing main menu to include icons to access the functional areas shown below:

* **Mortgage calculator** function: (i.e. Loan repayment calculator)
* **Currency converter** function: (i.e. Foreign exchange calculator)

Your team needs to nominate a team leader. The team leader needs to prepare the local repository and ensure the existing code is built without errors before its synchronized to the remote repository. All team members will assist in the preparation process. This will include:

* Customise the version control workflow
* Build the application in the VCS
* Test the application in VCS
* Write an installation disruption plan.
* Seek approval for the change overall
* Each student must clone the remote repository into the local working folder for development.

**Requirements:**

**Team arrangement:**

*The assignment is a practical exercise which requires members to work together in a team development environment.*

* *Each team must be no more than 2 members.*
* *Organise your team so that you have a team leader (administrator) and a developer member.*
* *All students must register an account with GitHub.*
* *A GitHub project name* ***Tafexxxxyy*** *will be assigned by your lecturer.* ***xxxx*** *is the semester name such as 22S1 and* ***yy*** *is a team number such as* ***01, 02, 03*** *…. etc.*
* *A team leader will be nominated by team members or assigned by your lecturer. The team leader will create the GitHub project.*
* *The GitHub project team leader will invite the other team members and your lecturer (e.g. tafesakt) as the collaborators.*
* *The initial program for loading into repository is available from the* ***ICTICT449 - ASDS - Assignment (Universal Calculator).zip*** *file.*

***For Submission:***

* *The oral communication assessment is conducted in Week 3 and the written VCS requirements report is due by Week 4.*
* *The assignment has individual assessment and team assessment components. All individual and team submissions are due by Week 7.*

***Team Work:*** *Team Leader must submit the Team work on behalf of the other team members through the Team Work upload submission link in LEARN.*

* *Submit the* ***final version of the C# code*** *checked out from the master branch for marking.*
* *The team leader is responsible for uploading the files & report for all of the team work.*
* In the screen shots, circle your answer in red.
* The final report must include all team member names and their roles.
* The final team report must have the name of the GitHub project name.
* In the GitHub, you must add the lecturer and the team members as contributors.

**Individual Work:** Each team member needs to submit through the Individual Work upload submission link.

* S*ubmit the* ***final version of the C# code*** *checked out on your feature branch for marking.*
* A Document report that must include the answers and screen shots for each individual task. *Capture screen shots during your development with branching & merging.* In the screen shots, circle your answer in red.
* Include **ITWorkds Version Control System Test Outcomes document**
* Include **ITWorks Service Disruption Notification document.**
* When displaying the revision graph, make sure that you include the option of all branches.

**SECTION A. (Team Work – Oral and Written CVS requirements)**

**VCS requirements**

The ITWorks manager wants the software development to staff to implement the version control guidelines & workflow practices (I.e. DevOps practices) in the organization. To decide what VCS system to use, the team must conduct a meeting with the team members to discuss the VCS requirements for developing the application and the requirements report after the meeting.

ITWorks expects team members to follow the communication protocols which are listed in **ITWorks Communication Protocols.docx**. While the team is conducting the meeting, your lecturer will act as the IT Manager and to sign off the team members has met those protocols specified as the following:

* Raise questions to seek for clarification in the context of VCS project. (at least two questions)
* Respond to the questions being asked by the other team members. (at least two responses)
* Engage into one-to-one conversation (at least two conversations per each member)
* Recap and draw conclusion from the meeting (summarise the discussion)
* Close the meeting (Document the discussion)

In the meeting, the team members must discuss the following VCS requirements item. After the meeting, the team will document VCS requirements to conclude the outcomes from the meeting.

1. There are two kinds of version control systems (VCS), namely distributed VCS and Centralised VCS. Conclude the discussion by listing two differences between these version control systems.

|  |  |
| --- | --- |
| Distributed version control system | Centralised version control system |
| 1. Every developer has its own exact replica of the copy and has the full history on their hard drive | 1.Has one central repository on a server where client will have a working version and multiple developers can pull and push changes to the central copy |
| 2. Doesn’t need to be connected to network do get changes done and until network is connected you can push and commit changes to central repository | 2. Must be connected to network otherwise you wouldn’t be able to do anything |

1. The team must name two products for each of these version control systems available in the market.

|  |  |
| --- | --- |
| Distributed version control system | Centralised version control system |
| 1.Git | 1.Subversion (SVN) |
| 2.Mecurial | 2.Perforce |

1. The team must choose which VCS product to develop the Universal Calculator project and justify with two reasons why the team made this decision.

Which VCS product to be used?

Between Git and Mercurial as they are both very similar in terms of how they are able to handle small to very large projects they both are quick in performance, but even though Mercurial is simpler and easier to understand. does have a bit of a flaw even though it is safer for less experienced users, it doesn’t allow you to change history. However, git allows all involved developers to change the version history. This could have potential consequences when working as mercurial only lets you change your last commit

Two reasons are:

* Git allows you to change any version of history and does not have to be the latest
* Branch in Mercurial cannot be removed because that would alter the history. In Git it allows you to create, delete, and change a branch anytime without affecting the commits and has a staging area before adding files for your next commit (useful if you don’t want to commit all changed files together).

1. The team must suggest aclient tool )i.e. product) for interfacing the repository for the VCS product chosen in part c. Give two reasons why you choose them.

Which client Tool? TortoiseGit

Two reasons are:

* It’s an open source
* It has a git so it is functional without needing commands

1. With the provided **ITWorks Git Workflow guidelines**, read through the whole assignment what you need to do. Design the version control workflow about how your team would accomplish the development, e.g.:

* What branch name to be used for the initial loading of codes

Pascal case convention ie. SalesAppRepos

* What branch name to be used for what feature

feature\_mortgage and feature\_currency\_conversion

* Which feature branch to merge to the development branch first, then second & third.... until the product release. Merge into development both go in dev and gebnerate another version when merguing
* What tag to be used for the product release

To document your design, draw the Git workflow for your project.

|  |
| --- |
|  |

Workflow is still missing some tages for the development versions before we merge in the features - they'll be V0.1, V0.2, or could go V0.1.0 then 0.1.1 etc

**SECTION B. (Team Work – Configure the VCS)**

In using the VCS in the development of the application, start by setting up the local repository.

For submission, **each** of the following steps should, capture a series of screen shots for creating and configuring both local and remote repositories. Make sure the screen shots are pasted into the final report for submission. Label the task number and the name of the screens clearly.

1. **Setting up the local repository:**

* Creates a local working directory and create a local Git repository called **Tafexxxxyy**. (The name will be allocated by the lecturer).
* Add the given **.gitignore**, **README.md**, **LICENSE.md** files and commit to the local repository.
* Copy the C# calculator files into the repository and the group leader ensure the file is running with no error. Add the files into the staging environment. Commit the C# codes with appropriate comment message “Initial loading” which reflects the change activities.
* Use the show log to show the status of the repository.

For submission, capture the following screen shots:

* the .gitignore, README.md and LICENSE.md file in the local Repository.
* the list of C# code file added to the staging environment
* the list of files committed to the repository (i.e. files with green tick overlay icons)
* In the show log showing the log of the commits with the comments as “Initial loading”

1. **Setting up the remote repository**

* Each team need to setup the project in GitHub as **Tafexxxxyy.**

Record your GitHub repository URL address here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Create a GitHub repository with the default branch name as “**main**”.
* Add the **Discussion** feature to be used in the repository for member collaboration.
* The Team leader must invite your lecturer (**tafesakt**) as one of your collaborators.
* Add all contributors to the GitHub project as Collaborator. (Note: the collaborator needs to accept the invitation).
* Synchronise the local repository with the GitHub repository.

Submission Requirements: Capture the following screen shots of:

* the GitHub repository created
* the Discussion feature is included in the GitHub Repository
* the list of collaborators accepted the invitation
* the synchronization done (i.e. the .gitignore, README, License files and the C# files are in the “main’ branch of the remote repository)

1. **Prepare the development branch before major development**

* Based on the head revision of the local master branch, create a development branch.
* Checkout the development branch, the team leader designs the **main menu** (see Appendix) with the XAML code. Add a text block with the text as “**Universal Calculator**” and four buttons (**Math Calculator, Mortgage calculator, Currency Calculator** and **Exit**). No need to write the C# code for the button click event yet. Change the application to display the main menu first when the application is executed. Add the files that has changes of codes into the staging environment. Note: do not commit now.
* In the main menu page, add a logo image (abacus.png available from the student files) to the top right hand corner.
* Add the files with the changes again into the staging environment. Note: do not commit now.
* Show the log of the **changes** which are in the staging area. Provide a screen shot of those file changes such as modified and added.
* Commit the changes in the development branch to give a new revision number with appropriate comments. Show the revision number of the development branch of the repository. Provide a screen shot of the revision number after the commit.
* Push the changes of all branches in the local repository to the remote repository. Provide a screen shot of the log of files in the remote development branch. (Note: To do this you need to configure the TortoiseGit Putty Generator with the public/private key pairs and upload

the public key ring to the GitHub project.)

Your submission must capture the following screen shots:

* The development branch has been created.
* The .XAML and .CS file names are staged in the development branch.
* Show the difference of the code in the .XAML and .CS files (use show log with diff)
* Show the .XAML and .CS files committed in the development branch

**SECTION C. (Individual work) -** Develop features using branching and merging

\*\*note: Start this section only after all tasks in Section B are completed.

1. **Configure the client tool to interface with the remote repository**
2. Each individual member clone the GitHub repository to a local working repository.

* Each individual member configures the TortoiseGit setting for the network connection to the GitHub repository with the encryption/decryption techniques using the private/public key pairs.
* Individual member (i.e. the collaborator) needs to give the public key to the group leader to add into the key ring of the GitHub repository. The individual member must keep passphrase and the private key for future use.

Your submission should capture the following screen shots:

* The TortoiseGit settings with the global .gitconfig with your name and email & local .git/config details
* The TortoiseGit remote settings with the remote URL to the GitHub and Putty key settings

1. **Communicating using the collaborative tools**

Each individual member is responsible for the development of one of the following functions in a feature branch simultaneously:

|  |  |
| --- | --- |
| **Feature branches:** | **Features to develop:** |
| Mortgage calculator  (Team member 1) | i.e. Loan repayment calculator:   * Develop the XAML page for capturing user input and display output. See Appendix for the page layout. * Write C# code to process the input data in order to produce the output. * The calculation formula (as shown in the Appendix) must be used inside a method which needs to pass the input parameter and return the result to screen. |
| Currency conversion calculator  (Team member 2) | i.e. Foreign exchange calculator:   * Develop the XAML page for capturing user input and display output. See Appendix for the page layout. * Write C# code to process the input data in order to produce the output. * The calculation formula (as shown in the Appendix) must be used inside a method which needs to pass the input parameter and return the result to screen. |

* Team members must use the collaboration tools (“Discussion”) provided in GitHub to communicate with each other. While developing your code individually, each member must raise three questions for discussion, and you must also respond to the discussions raised by the other member.

For submission, you need to provide evidence for your communication by capturing the following screen shots:

* A screen with the 3 discussions initiated by the team member
* A screen with the 3 responses to discussions raised by the team member

1. **Merge feature branches to development branch**

For developing the function in the feature branch:

* Checkout the head revision of the development branch, e.g. :
* Team member 1 creates the mortgage calculator branch.
* Team member 2 creates the currency conversion calculator branch
* The team members must develop their feature branches simultaneously. Individual team member works on his/her own feature branch. Make sure that there are no errors in the program. Capture a screen shot showing the feature branch (Mortgage calculator and the Currency Conversion calculator) during execution. Each team member commit the code to the feature branch with a comment. Push the feature branch to the remote repository. Capture a screen shot showing the revision graph with the checkout of the remote feature branch.
* Each team member will merge the feature branches into the development branch, one at a time. The merging process may involve the following steps:
* Checkout and pull the development branch to ensure the latest cod from the remote repository, since the other may have merged the codes before you.
* Merge from the feature branch into the development branch.
* Resolve conflicts (if any)
* Commit the changes to the local development branch with appropriate comments to reflect the development being done.
* Push the code to the remote development branch.
* The Team member must communicate with the other team members before and after the merge is carried out.
* Provide a revision graph with the feature branch checkout.
* A Team member must test the code to ensure that the development branch program is working after the merge. Provide screen shots to show your testing data works after your merge.

Your submission must capture the following screen shots:

* After the commit and but before the merge, checkout the latest revision of your feature branch. Show the revision graph.
* Before the merge, show the code differences with the previous revision in the show log.
* After the merge, display the revision graph to ensure the merge was done.
* After the merge, use the show log to show that the code has been merged into the development branch.
* Capture the screen shots that the program is running with the features after the merge.

**SECTION D. (Team Work) –** Preparing for product release

* After all branches have been merged into the development branch, merge the development branch to the release branch as follows:
* Create a release branch based on the development branch.
* Merge the development branch into the release branch.
* Show the revision graph with the check-out of the release branch

Your submission must capture the following screen shot:

* The revision graph with the checkout of the release branch
* Do a final check on the latest revision of the release branch to ensure it has got all the required features. Merge the release branch into the master branch using following steps:
* Checkout the release branch.
* Completed the coding for the navigation between the pages such as the Main Menu page may call the mortgage calculator page, the currency conversion calculator page, and the simple calculator page.
* Commit all changes to the release branch.
* Merge the release branch into the master branch.
* Show the revision graph with the check-out of the master branch
* Push the master branch to the remote main branch

Your submission must capture the following screen shots:

* Show the overlay icons with those files being modified before committing to the release branch.
* The Revision graph with the checkout of the master branch.
* Push all branches to the remote repository.
* Checkout the master branch with the latest revision and create a tag with name V2.0 using following steps:
* Show the log of the repository.
* Checkout the tag V2.0
* Show the Revision graph.

Your submission must capture the following screen shot:

* Show the tags on the master branch.

**SECTION E. (Individual Work) –** Document the test outcomes

As the application Universal calculator v2.0 has been developed, you are asked to document and test the outcome of the revision numbers, and whether they match the major workflow. Use the given **ITWorks Version Control System Test Outcomes Document template.docx** to complete this task.

**Note**: All items in red must be replaced with Black coloured fonts and the description must be related with the scenario. You must delete the word ‘Template’.

Submit the **ITWorks Version Control System Test Outcomes Document** and obtain the IT Team leader (Your lecturer) for feedback. If the lecturer feedback is unsatisfactory you are required to respond accordingly to it. Otherwise, you will regard the final test outcomes are signed off by the IT Manager.

**SECTION F. (Individual Work)** – Create an installation disruption notification

Since the application Universal calculator v2.0 has been developed, ICT services needs you to prepare a software release i.e. to uninstall the older calculator v1.0 and install the newer v2.0 on staff workstations. You will need to notify all staff who will be affected by the installation disruption. Your notification should explain your deployment plan re the installation of the newer version. Use the **ITWorks Service Disruption Notification template.docx** to fill the details as required.

Note: All items in red must be replaced in black coloured fonts with the information related with the context of the scenario. You must delete the word ‘Template’.

You are required to submit the filled **ITWorks Service Disruption Notification document.**

**SECTION G. (Team Work)** – Respond to pull request

The repository of the Universal Calculator is owned by the Team Leader in the GitHub repository. The open-source development environment allows a public community developer to fork the repository to further develop the application.

* The other team member forks the repository to a new project.
* Check out the Master branch.
* Add another button called “Trip Calculator” to the main menu using XAML code.
* In the click event, add a message “Trip calculator C# code will be developed later”. No need to further develop the C# code for the click event for now.
* Raise a pull request to the original repository.
* The team leader responds to the pull request by accepting the request.
* Incorporate the extra codes.
* The team leader clones the repository and test the code showing the code has been added.

In your submission here, provide the following screen shots for all the above steps.

* + The screen layout added with the Trip Calculator button.
  + The pull request has been raised by the team member
  + The team leader got the notification of the pull request.
  + After cloning the repository, show the execution of the extra code being added to the repository.

***Submission Requirements:***

***Team Work***

*The**Team Leader must submit the Team work on behalf of the team through the Team Work upload submission link.*

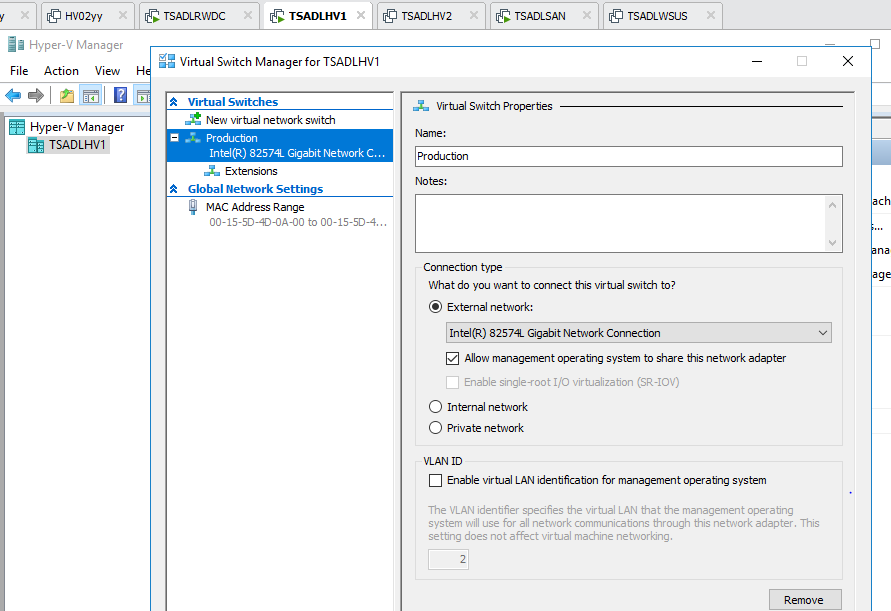
* *Submit the* ***final version of the C# code-*** *checked out from the master branch for marking.*
* *The team leader is responsible for uploading the files & report for all of the team work.* In your screen shots, circle your answers in red.
* The final report must include all team member names and their roles.
* The final team report must have the name of the GitHub project.
* In the GitHub, make sure that the contributors include your lecturer and the team members in it.

**Individual Work**

Each team member must submit through the Individual Work upload submission link.

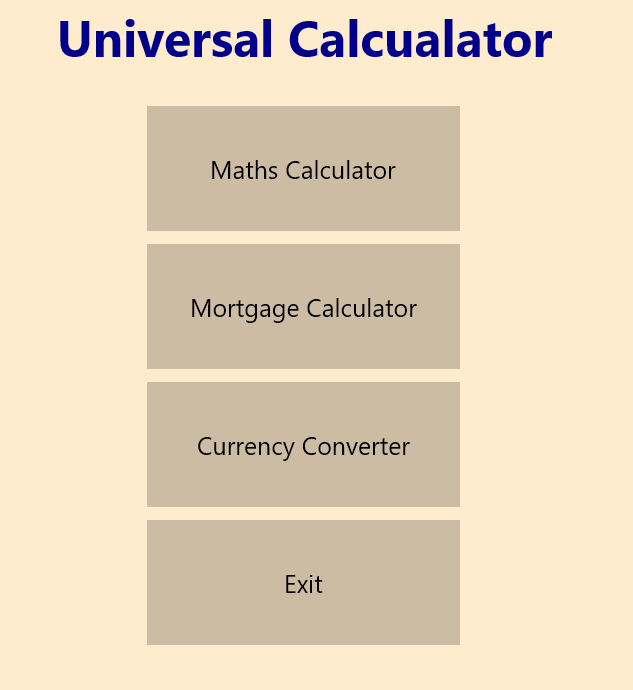
* S*ubmit the* ***final version of the C# code-*** *checked out on your feature branch for marking.*
* Document report must include the answers and screen shots for each individual task. *Capture screen shots during your development with branching & merging.* In your screen shots, circle your answer in red.
* Include **ITWorks Version Control System Test Outcomes document**
* Include the **ITWorks Service Disruption Notification document.**
* When displaying the revision graph, make sure that you include the option of all branches.

***Note:*** *When using screen captures where certain aspects of the image are required , you must identify them in RED - for example, using a red circle as shown in image below.*

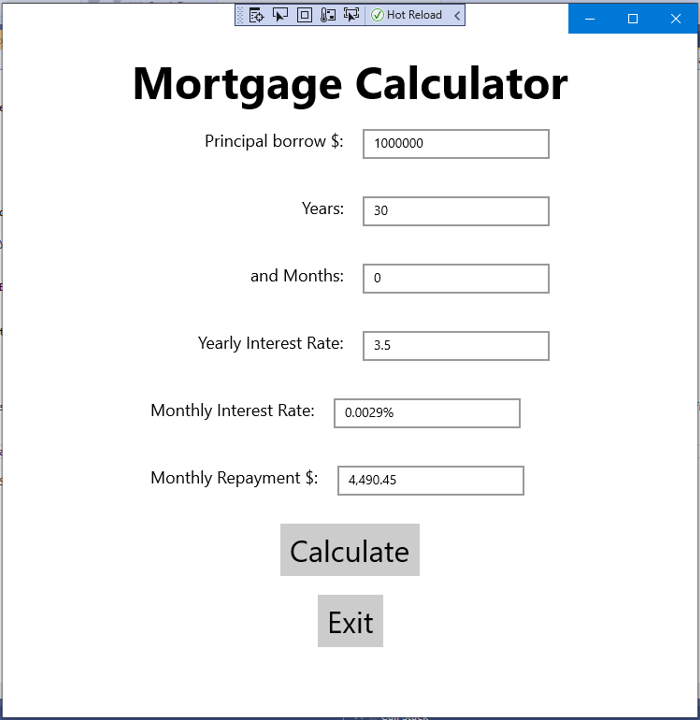


**Appendix:**

**The Main menu:**



**Mortgage calculator:**



Use the following equation for calculating the repayment:

**M = P [ i(1 + i)^n ] / [ (1 + i)^n – 1]**

P = principal loan amount

i = monthly interest rate

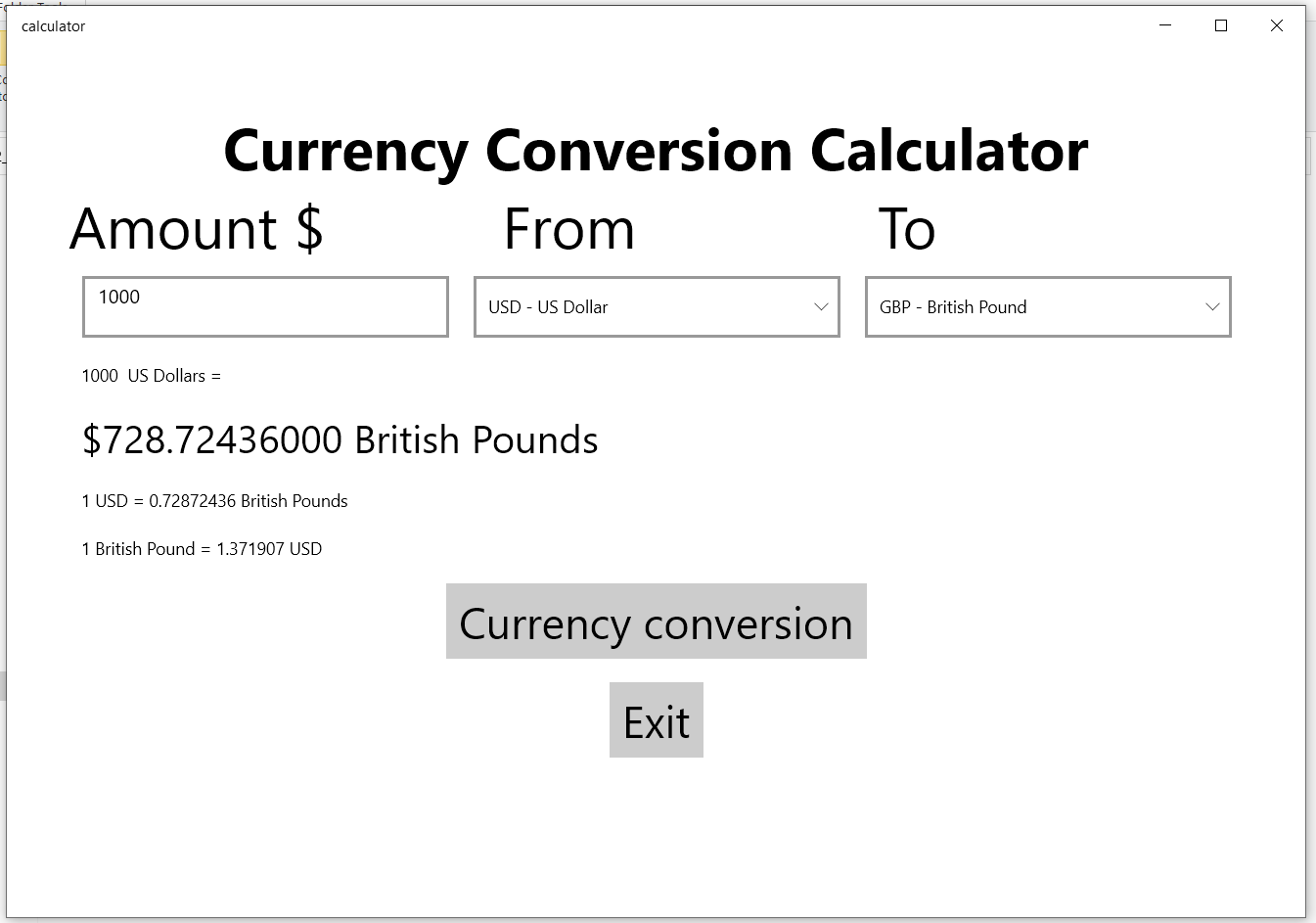
n = number of months required to repay the loan

For example, if the annual interest rate is 4%, the monthly interest rate would be 0.33% (0.04/12 = 0.0033).

A 30-year mortgage would require 360 monthly payments, while a 15-year mortgage would require exactly half that number of monthly payments, or 180.

Hints: use C# Math.Pow() function for calculating exponential.

**Currency Conversion calculator:**



Use the following conversion rate for testing your calculation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **From** | **To** | **Conversion Rate** |
| $1.0 | US Dollar | Euro | 0.85189982 |
|  |  | British Pound | 0.72872436 |
|  |  | Indian Rupee | 74.257327 |
| € 1.0 | Euro | US Dollar | 1.1739732 |
|  |  | British Pound | 0.8556672 |
|  |  | Indian Rupee | 87.00755 |
| £ 1.0 | British | US Dollar | 1.371907 |
|  |  | Euro | 1.1686692 |
|  |  | Indian Rupee | 101.68635 |
| ₹ 1.0 | Indian Rupee | US Dollar | 0.011492628 |
|  |  | Euro | 0.013492774 |
|  |  | British Pound | 0.0098339397 |