DeepLingo

Course: CPE009B
Academic Year/Semester: 2nd Year Semester

Team members:
Alcantara, Jason P.
Anastacio, Lester Arvid P.
Avila, Vince Gabriel V.
Catungal, Jan Kerwin B.

Submitted to:
Engr. Jimlord Quejado

Technological Institute of the Philippines

TABLE OF CONTENTS

THE PROBLEM

1.
Problem:
Lack of Exposure

Teenagers might not be exposed to a wide variety of reading materials or diverse vocabularies that would help them encounter new words and see how synonyms and antonyms are used in different contexts. Limited reading habits can contribute to limited word knowledge.

Impact:
Reduced Critical Thinking

A strong vocabulary is directly tied to critical thinking skills. If teenagers struggle with language nuances, it can hinder their ability to analyze complex ideas, identify key differences, and make informed decisions.

2.
Problem:
Dependence on Technology

Due to growth in digital communication (texts, social media, etc.) teenagers may become more dependent on abbreviations, slang, and informal language.

Impact:
Limited exposure to a variety of reading materials

Teenagers often aren't challenged with a range of reading materials that force them to be mindful of their vocabulary and comprehension. They often gravitate towards reading content that is easily consumable, such as social media, memes, or entertainment, that does not require them to think deeply or use complex vocabulary.

OBJECTIVES

• Create a working dictionary system that allows users to search for word meanings and translations easily.
• Apply programming and interface design concepts in developing the system.
• Improve teamwork, problem-solving, and communication within the group during the development process.
• Use organized coding structures and design patterns to ensure a smooth and efficient system flow.

RATIONALE
The DeepLingo Dictionary application is using a contiguous container which is the QVector<WordEntry> in which is located within a WordStorage singleton as the main in-memory data structure and such a decision ensures very good cache locality and an amortized O(1) time complexity for adding new records and although the UI layer manipulates entities like QListWidget for visualization, the core data model is based on QVector and the manner in which search and retrieval operations are presently implemented such as the Function::searchWord along with the filtering routines (wordsForLetter), necessitates a linear scan of all O(n) words in the store, thus the time complexity of word lookups is O(n). The space complexity of these operations is O(n + s) with being the temporary storage for the result lists, while the persistent memory is proportional to the total number of characters across all the WordEntry fields O(total characters stored).

CODES AND OUTPUT

In this section provides the complete codes and output of the app DeepLing, which is implemented using the programming language of c++ and the libraries of Qt and Cmake, on the first part of this section provides all the .cpp and header files and afterwards is the output of the entire application.

```
#include "Qt_includes.h"
#include "GUI/Gui_Holder.h"
#include "User_Files/UserStorage.h"
#include "User_Files/UserDialog.h"
#include "GUI/LoadingScreen.h"

int main(int argc, char *argv[]) {
    QApplication a(argc, argv);

    // Load existing users (file is users.json in cwd).
    // If a previous user was saved, they will be set as the current 
user here.
    UserStorage::instance().load("users.json");

    // Show the custom loading screen with title and animated bar.
    LoadingScreen loader;

    QEventLoop loop;
    QObject::connect(&loader, &LoadingScreen::finished, &loop, 
&QEventLoop::quit);
    loader.start(1400);
    loop.exec();

    // Show a modal dialog to collect user name and age (user can
cancel).
    // Only show the UserDialog if no user is currently set.
    if (UserStorage::instance().currentUser().isEmpty()) {
        UserDialog dlg;
        // The dialog handles saving the new user and setting them as
the current user on accept.
        dlg.exec();
    }

    // Show main application window.
    Gui_Holder w;
    w.show();

    return a.exec();
}
```

Figure 1: **main.cpp**

```
#ifndef QT_INCLUDES_H
#define QT_INCLUDES_H

// --- Qt Core/Base Headers ---
#include <QApplication>
#include <QChar>
#include <QDate>
#include <QDebug>
#include <QEventLoop>
#include <QIcon>
#include <QJsonObject>
#include <QJsonArray>
#include <QJsonDocument>
#include <QString>
#include <QStringList>
#include <QVector>
#include <QDir>
#include <QFileInfo>
#include <QColor>
```

```
#include <QFile>
#include <QStyle>

// --- Qt Widget/GUI Headers ---
#include <QWidget>
#include <QMainWindow>
#include <QDialog>
#include <QLabel>
#include <QPixmap>
#include <QTabWidget>
#include <QLineEdit>
#include <QTextEdit>
#include <QComboBox>
#include <QPushButton>
#include <QCloseEvent>
#include <QMessageBox>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QGridLayout>
#include <QScrollArea>
#include <QFont>
#include <QSplitter>
#include <QListWidget>
#include <QListWidgetItem>
#include <QCalendarWidget>
#include <QSplashScreen>
#include <QTimer>
#include <QDialogButtonBox>
#include <QSpinBox>

// --- Qt Graphics/Painting Headers ---
#include <QPainter>
#include <QPen>
#include <QBrush>

// --- Project Headers (Included for PCH benefit) ---
#include "User_Files/User.h"
#include "User_Files/UserStorage.h"
#include "Word_Files/Word_Storage.h"
#include "Function_Files/Function.h"

#endif // QT_INCLUDES_H
```

Qt_Includes.h

```
#include "Function_Files/Function.h"
#include "Word_Files/Word_Storage.h"
#include "User_Files/UserStorage.h"
#include "Qt_includes.h"

QChar Function::normalizeKey(const QString &word) const {
    if (word.isEmpty()) return QChar('\0');
    QChar c = word.at(0);
    if (!c.isLetter()) return QChar('\0');
    return c.toLower();
}

int Function::addWord(const QString &word, const QString &definition,
const QString &translation) {
    if (word.trimmed().isEmpty() || definition.trimmed().isEmpty())
return 1;
    QChar key = normalizeKey(word);
    if (key == QChar('\0')) return 1;

    auto list = WordStorage::instance().wordsForLetter(key);
    for (const auto &e : list) {
        if (e.word.compare(word, Qt::CaseInsensitive) == 0) return 3;
// duplicate
    }

    const int MAX_PER_LETTER = 30;
    if (list.size() >= MAX_PER_LETTER) return 2;

    WordEntry entry;
    entry.word = word.trimmed();
    entry.definition = definition.trimmed();
    entry.translation = translation.trimmed();
    WordStorage::instance().addWord(entry);
    WordStorage::instance().save();
    return 0;
}

QString Function::searchWord(const QString &word, bool
getTranslation) const {
    if (word.trimmed().isEmpty()) return QString();
    auto all = WordStorage::instance().allWords();
    for (const auto &e : all) {
        if (e.word.compare(word, Qt::CaseInsensitive) == 0) {
            return getTranslation ? e.translation : e.definition;
        }
    }
    return QString();
}

QVector<QPair<QString, QString>> Function::getWordsByLetter(QChar
letter, bool getTranslation) const {
    QVector<QPair<QString, QString>> out;
    auto list = WordStorage::instance().wordsForLetter(letter);
    out.reserve(list.size());
    for (const auto &e : list) {
        out.append(qMakePair(e.word, getTranslation ? e.translation :
e.definition));
    }
    return out;
}

bool Function::addWordEntry(const WordEntry &entry)
{
    WordStorage::instance().addWord(entry);

    QString currentUserName = UserStorage::instance().currentUser();
    if (!currentUserName.isEmpty()) {
        QJsonObject userData =
UserStorage::instance().currentUserData();
        QStringList addedWordsList;
        QJsonValue addedWordsValue = userData.value("addedWords");

        if (addedWordsValue.isArray()) {
            QJsonArray addedArray = addedWordsValue.toArray();
            for (const QJsonValue &v : addedArray) if (v.isString())
addedWordsList.append(v.toString());
        }

        if (!addedWordsList.contains(entry.word,
Qt::CaseInsensitive)) {
            addedWordsList.append(entry.word);

            QJsonArray newAddedWordsArray;
            for (const QString &w : addedWordsList)
newAddedWordsArray.append(w);
            userData["addedWords"] = newAddedWordsArray;

            UserStorage::instance().updateCurrentUserData(userData);
            UserStorage::instance().saveCurrentUserData();
        }
    }

    return WordStorage::instance().save();
}

bool Function::addWordFromInputs(const QString &word,
                                 const QString &definition,
                                 const QString &translation,
                                 const QString &synonymsCsv,
                                 const QString &antonymsCsv,
                                 const QString &background,
                                 const QString &usage)
{
    if (word.trimmed().isEmpty()) return false;

    WordEntry e;
    e.word = word.trimmed();
    e.definition = definition.trimmed();
    e.translation = translation.trimmed(); // <-- add Tagalog support

    auto splitCsv = [](const QString &s) -> QStringList {
        QStringList parts;
        for (const auto &p : s.split(',', Qt::SkipEmptyParts))
parts.append(p.trimmed());
        return parts;
    };

    e.synonyms = splitCsv(synonymsCsv);
    e.antonyms = splitCsv(antonymsCsv);
    e.background = background.trimmed();
    e.usage = usage.trimmed();

    return addWordEntry(e);
}
```

Function.cpp

```
#ifndef FUNCTION_H
#define FUNCTION_H

#include <QString>
#include <QVector>
```

```
#include <QPair>
#include "Word_Files/Word_Storage.h"

class Function {
private:
    QChar normalizeKey(const QString &word) const;

public:
    Function() = default;

    // Updated signatures with translation support
    int addWord(const QString &word, const QString &definition, const
QString &translation = QString());

    QString searchWord(const QString &word, bool getTranslation =
false) const;
    QVector<QPair<QString, QString>> getWordsByLetter(QChar letter,
bool getTranslation = false) const;

    bool addWordEntry(const WordEntry &entry);
    bool addWordFromInputs(const QString &word,
                           const QString &definition,
                           const QString &translation,
                           const QString &synonymsCsv,
                           const QString &antonymsCsv,
                           const QString &background,
                           const QString &usage);
};

#endif // FUNCTION_H
```

Function.h

```
#include "GUI/AboutWindow.h"
#include "Qt_includes.h"

// Constructor initializes the UI.
AboutWindow::AboutWindow(QWidget *parent)
    : QDialog(parent)
{
    setWindowTitle(tr("About DeepLingo"));
    setMinimumSize(450, 450);
    setModal(true);
    // Apply the same green styling for consistency
    setStyleSheet(R"(
        QDialog {
            background-color: #f0f2f5;
        }
        QLabel {
            color: #4CAF50; /* Green title */
            font-size: 24px;
            font-weight: bold;
        }
        QTextEdit {
            border: 1px solid #d0d0d0;
            border-radius: 8px;
            padding: 10px;
            background-color: #ffffff;
        }
    )");

    setupUI();
    populateContent();
}

AboutWindow::~AboutWindow() {}

// Sets up the visual structure (labels, layout).
void AboutWindow::setupUI()
{
    QVBoxLayout *mainLayout = new QVBoxLayout(this);
    mainLayout->setContentsMargins(20, 20, 20, 20);

    // Title Label
    titleLabel = new QLabel(tr("Project DeepLingo"), this);
    // No explicit alignment on the label itself, let the layout
center it.

    // Create a horizontal layout to truly center the titleLabel
    QHBoxLayout *titleLayout = new QHBoxLayout();
    titleLayout->addStretch();
    titleLayout->addWidget(titleLabel);
    titleLayout->addStretch();

    mainLayout->addLayout(titleLayout); // Add the horizontal layout
to the main vertical layout
    mainLayout->addSpacing(15);
    // Content Display Area (where members/credits will go)
    contentDisplay = new QTextEdit(this);
    contentDisplay->setReadOnly(true);
    mainLayout->addWidget(contentDisplay);
}

// Populates the QTextEdit with the About content.
void AboutWindow::populateContent()
{
    // *** Text that you'll see in the about window ***
    QString content = R"(
        <p style="font-size: 14px; color: #333;">
        Welcome to DeepLingo, your personalized dictionary and
vocabulary builder.
        </p>

        <h3 style="color: #4CAF50;">Project Development Team</h3>
        <p>
        This application was proudly created as a collaborative
effort by the following members:
        </p>

        <ul>
            <li><strong>Anastacio, Lester Arvid P.:</strong> [Role -
e.g., Lead Developer]</li>
            <li><strong>Catungal, Jan Kerwin B. 2:</strong> [Role -
e.g., Documentation/UI/UX Design]</li>
            <li><strong>Avila, Vince Gabriel V:</strong> [Role -
e.g., Documentation/Words For English and Tagalog]</li>
            <li><strong>Alcantara, Jason P:</strong> [Role - e.g.,
Documentation/Words For English and Tagalog]</li>
        </ul>
        <!-- -->)";
    // ****************************************************

    contentDisplay->setHtml(content);
}
```

AboutWindow.cpp

```
#ifndef ABOUTWINDOW_H
#define ABOUTWINDOW_H
```

```
#include <QDialog>
#include <QLabel>
#include <QTextEdit>
#include <QVBoxLayout>

class AboutWindow : public QDialog
{
    Q_OBJECT

public:
    // Constructor. Takes the parent widget for proper modal behavior.
    explicit AboutWindow(QWidget *parent = nullptr);
    ~AboutWindow();

private:
    QLabel *titleLabel;
    QTextEdit *contentDisplay;

    // Method to set up all widgets and layout.
    void setupUI();
    // Method to populate the editable content.
    void populateContent();
};

#endif // ABOUTWINDOW_H
```

AboutWindow.h

```
#include "Qt_includes.h"
#include "GUI/Gui_Holder.h"
#include "GUI/UserProfileWindow.h"
#include "GUI/AboutWindow.h"
#include "GUI/WordDetailWindow.h"
#include "Function_Files/Function.h"
#include "User_Files/UserStorage.h"
#include "Word_Files/Word_Storage.h"
#include "User_Files/User.h"
#include <QListWidget>


// Helper function to extract initials from a full name.
static QString initialsFromName(const QString &name) {
    QStringList parts = name.simplified().split(' ', Qt::SkipEmptyParts);
    if (parts.isEmpty()) return QString();
    QString out;
    out.append(parts.value(0).left(1).toUpper());
    if (parts.size() > 1) out.append(parts.value(1).left(1).toUpper());
    return out;
}

// Constructor: Sets up the main window UI and initializes the function pointer.
Gui_Holder::Gui_Holder(QWidget *parent)
    : QMainWindow(parent),
      m_appFunction(new Function) // Initialize the Function pointer
{
    setWindowTitle("DeepLingo");
    setMinimumSize(700, 520);

    // Global UI styling (Green Theme).
    setStyleSheet(R"(
        QMainWindow {
            background-color: #f0f2f5;
            font-family: 'Segoe UI', sans-serif;
        }
        QLabel {
            color: #333333;
        }
        QLineEdit, QTextEdit, QComboBox {
            border: 1px solid #d0d0d0;
            border-radius: 8px;
            padding: 8px 10px;
            background-color: #ffffff;
            selection-background-color: #a8e6cf;
        }
        QLineEdit:focus, QTextEdit:focus, QComboBox:focus {
            border: 1px solid #4CAF50;
            outline: none;
        }
        QPushButton {
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 8px;
            padding: 10px 15px;
            font-weight: bold;
        }
        QPushButton:hover {
            background-color: #45a049;
        }
        QPushButton:pressed {
            background-color: #3e8e41;
        }
        QTabWidget::pane {
            border: 1px solid #d0d0d0;
            border-radius: 8px;
            background-color: #ffffff;
            margin-top: -1px;
        }
        QTabBar::tab {
            background: #e0e0e0;
            color: #555555;
            padding: 10px 15px;
            border: 1px solid #d0d0d0;
            border-top-left-radius: 8px;
            border-top-right-radius: 8px;
            margin-right: 2px;
        }
        QTabBar::tab:selected {
            background: #ffffff;
            color: #4CAF50;
            border-bottom-color: #ffffff;
        }
        QTabBar::tab:hover {
            background: #f0f0f0;
        }
    )" );


    setupUI();

    // Ensure words are loaded from storage.
    WordStorage::instance().load();

    // Update the profile button's appearance.
    updateProfileView();
}

// Destructor.
Gui_Holder::~Gui_Holder() {
    delete m_appFunction; // Delete the Function object
}

// Initializes all widgets and layouts.
void Gui_Holder::setupUI()
{
    QWidget *cent = new QWidget(this);
    QVBoxLayout *mainLay = new QVBoxLayout(cent);
    mainLay->setContentsMargins(15, 15, 15, 15);

    // Top Bar: Logo and Button Container
    QWidget *topBar = new QWidget(cent);
    QHBoxLayout *topLay = new QHBoxLayout(topBar);
    topLay->setContentsMargins(0, 0, 0, 0);


    // Invisible spacer/stretch on the left to push the logo right
    topLay->addStretch();

    // Application title / Logo
    m_logoLabel = new QLabel(cent);
    if (m_logoImage.load("DeepLingo_Logo.png")) {
        m_logoLabel->setPixmap(m_logoImage.scaled(650, 160, Qt::KeepAspectRatio,
Qt::SmoothTransformation));
        m_logoLabel->setAlignment(Qt::AlignCenter);
    } else {
        // Fallback to text if image not found
        m_logoLabel->setText("<h1><b style='color:#4CAF50;'>DeepLingo</b></h1>");
        m_logoLabel->setAlignment(Qt::AlignCenter);
        qWarning("Gui_Holder: Title logo 'DeepLingo_Logo.png' not found. Falling back to text.");
    }
    m_logoLabel->setMaximumWidth(650);
    topLay->addWidget(m_logoLabel);

    // Invisible spacer/stretch in the middle to push the right buttons far right
    topLay->addStretch();

    // Vertical Container for Profile and Settings Buttons (Right-aligned)
    QWidget *rightButtonContainer = new QWidget(topBar);
    QVBoxLayout *rightButtonLay = new QVBoxLayout(rightButtonContainer);
    rightButtonLay->setContentsMargins(0, 0, 0, 0);
    rightButtonLay->setSpacing(5);

    // Profile Button (Top Right)
    profileButton = new QPushButton(rightButtonContainer);
    profileButton->setFixedSize(40, 40);
    profileButton->setIcon(this->style()->standardIcon(QStyle::SP_MessageBoxInformation));
    profileButton->setIconSize(QSize(24, 24));
    profileButton->setText("");
    profileButton->setToolTip(tr("Profile"));
    // Styling for the profile button.
    profileButton->setStyleSheet(R"(
        QPushButton {
            background-color: #4CAF50;
            border-radius: 20px;
            padding: 0px;
        }
        QPushButton:hover { background-color: #45a049; }
        QPushButton:pressed { background-color: #3e8e41; }
    )");
    rightButtonLay->addWidget(profileButton);
    connect(profileButton, &QPushButton::clicked, this, &Gui_Holder::on_profileButton_clicked);

    // Settings Button (Below Profile)
    settingsButton = new QPushButton(rightButtonContainer);
    settingsButton->setFixedSize(40, 40);
    settingsButton->setIcon(this->style()->standardIcon(QStyle::SP_FileDialogDetailedView));
    settingsButton->setIconSize(QSize(24, 24));
    settingsButton->setText("");
    settingsButton->setToolTip(tr("Settings / About"));
    // Styling for the settings button (same style).
    settingsButton->setStyleSheet(R"(
        QPushButton {
            background-color: #4CAF50;
            border-radius: 20px;
            padding: 0px;
        }
        QPushButton:hover { background-color: #45a049; }
        QPushButton:pressed { background-color: #3e8e41; }
    )");
    rightButtonLay->addWidget(settingsButton);
    connect(settingsButton, &QPushButton::clicked, this,
&Gui_Holder::on_settingsButton_clicked);

    // Add stretch to push buttons to the top of the container
    rightButtonLay->addStretch();

    // Add the button container to the main top bar layout
    topLay->addWidget(rightButtonContainer);

    mainLay->addWidget(topBar);
    mainLay->addSpacing(10);

    // QTabWidget setup
    QTabWidget *tabs = new QTabWidget(cent);

    // Setup the "Add Word" tab.
    QWidget *addTab = new QWidget;
    QVBoxLayout *addLay = new QVBoxLayout(addTab);
    addLay->setContentsMargins(15, 15, 15, 15);

    addLay->addWidget(new QLabel(tr("Word:"), addTab));
    wordInputAdd = new QLineEdit(addTab);
    addLay->addWidget(wordInputAdd);

    addLay->addWidget(new QLabel(tr("Definition:"), addTab));
    definitionInputAdd = new QTextEdit(addTab);
    definitionInputAdd->setMinimumHeight(80);
    addLay->addWidget(definitionInputAdd);

    addLay->addWidget(new QLabel(tr("Synonyms (comma separated):"), addTab));
    synonymsInput = new QLineEdit(addTab);
    addLay->addWidget(synonymsInput);

    addLay->addWidget(new QLabel(tr("Antonyms (comma separated):"), addTab));
    antonymsInput = new QLineEdit(addTab);
    addLay->addWidget(antonymsInput);

    addLay->addWidget(new QLabel(tr("Background / Etymology:"), addTab));
    backgroundInput = new QTextEdit(addTab);
    backgroundInput->setMinimumHeight(80);
    addLay->addWidget(backgroundInput);

    addLay->addWidget(new QLabel(tr("Usage / Example:"), addTab));
    usageInput = new QTextEdit(addTab);
    usageInput->setMinimumHeight(60);
    addLay->addWidget(usageInput);
    addLay->addStretch();

    addWordButton = new QPushButton(tr("Add Word"), addTab);
    connect(addWordButton, &QPushButton::clicked, this,
&Gui_Holder::on_addWordButton_clicked);
    addLay->addWidget(addWordButton);


    tabs->addTab(addTab, tr("Add Word"));

    // Setup the "Search" tab.
    QWidget *searchTab = new QWidget;
    QVBoxLayout *sLay = new QVBoxLayout(searchTab);
    sLay->setContentsMargins(15, 15, 15, 15);
    sLay->addWidget(new QLabel(tr("Search Word:"), searchTab));
    wordInputSearch = new QLineEdit(searchTab);
    sLay->addWidget(wordInputSearch);
```

```
   searchWordButton = new QPushButton(tr("Search"), searchTab);
   sLay->addWidget(searchWordButton);
   resultOutputSearch = new QTextEdit(searchTab);
   resultOutputSearch->setReadOnly(true);
   resultOutputSearch->setStyleSheet("background-color: #f8f8f8;");
   resultOutputSearch->setMinimumHeight(250);
   sLay->addWidget(resultOutputSearch);
   sLay->addStretch();
   connect(searchWordButton, &QPushButton::clicked, this,
&Gui_Holder::on_searchWordButton_clicked);
   tabs->addTab(searchTab, tr("Search"));

   // Setup the "Browse" tab.
   QWidget *browseTab = new QWidget;
   QVBoxLayout *bLay = new QVBoxLayout(browseTab);
   bLay->setContentsMargins(15, 15, 15, 15);

   letterComboBox = new QComboBox(browseTab);
   // Populate combo box with letters A-Z.
   for (char c='A'; c<='Z'; ++c) letterComboBox->addItem(QString(c));
   bLay->addWidget(letterComboBox);
   browseOutput = new QListWidget(browseTab);
   browseOutput->setStyleSheet("background-color: #f8f8f8;");
   browseOutput->setMinimumHeight(300);
   browseOutput->setSelectionMode(QAbstractItemView::SingleSelection);
   bLay->addWidget(browseOutput);
   bLay->addStretch();
   // Connect combo box signal to slot to update list.
   connect(letterComboBox, QOverload<int>::of(&QComboBox::currentIndexChanged),
         this, &Gui_Holder::on_letterComboBox_currentIndexChanged);

   // Connect list item click to open detail window
   connect(browseOutput, &QListWidget::itemClicked, this,
&Gui_Holder::on_browseItem_clicked);

   // Trigger the initial display of words starting with 'A'
   on_letterComboBox_currentIndexChanged(0);

   tabs->addTab(browseTab, tr("Browse"));

   mainLay->addWidget(tabs);

   setCentralWidget(cent);

   // Set initial avatar text.
   updateProfileAvatar();
```

```
}

// Refreshes the profile view.
void Gui_Holder::updateProfileView()
{
    updateProfileAvatar();
}

// Updates avatar appearance based on current user.
void Gui_Holder::updateProfileAvatar()
{
    QString user = UserStorage::instance().currentUser();
    if (user.isEmpty()) {
        profileButton->setToolTip(tr("No user selected"));
        // Icon for 'no user/add user'.
        profileButton->setIcon(this->style()->standardIcon(QStyle::SP_MessageBoxQuestion));
    } else {
        profileButton->setToolTip(user);
        // Icon for 'user profile'.
        profileButton->setIcon(this->style()->standardIcon(QStyle::SP_MessageBoxInformation));
    }
}

// Handles the Settings/About button click event.
void Gui_Holder::on_settingsButton_clicked()
{
    // Launch the AboutWindow modally.
    AboutWindow aboutDlg(this);
    aboutDlg.exec();
}
// ---------------------------------------------------------------

// Handles the Add Word button click event: processes input and saves to storage.
void Gui_Holder::on_addWordButton_clicked()
{
    QString w = wordInputAdd->text().trimmed();
    QString def = (static_cast<QTextEdit*>(definitionInputAdd))->toPlainText().trimmed();
    QString syn = synonymsInput->text().trimmed();
    QString ant = antonymsInput->text().trimmed();
    QString bg = backgroundInput->toPlainText().trimmed();
    QString usage = usageInput->toPlainText().trimmed();

    if (w.isEmpty()) {
        QMessageBox::information(this, tr("Missing"), tr("Please enter a word."));
        return;
    }
    bool ok = m_appFunction->addWordFromInputs(w, def, QString(), syn, ant, bg, usage);
    if (ok) {
        QMessageBox::information(this, tr("Added"), tr("Word added successfully."));

        // Clear input fields after successful addition.
        wordInputAdd->clear();
        definitionInputAdd->clear();
        synonymsInput->clear();
        antonymsInput->clear();
        backgroundInput->clear();
        usageInput->clear();
    } else {
        QMessageBox::warning(this, tr("Failed"), tr("Could not add word (maybe duplicate)."));
    }
}

// Handles the Search Definition button click event: retrieves and displays word details.
void Gui_Holder::on_searchWordButton_clicked()
{
    QString key = wordInputSearch->text().trimmed();
    if (key.isEmpty()) return;

    // Search for the word in storage.
    auto words = WordStorage::instance().allWords();
    for (const auto &e : words) {
        if (e.word.compare(key, Qt::CaseInsensitive) == 0) {
            QString out;
            out += "Word: " + e.word + "\n\n";
            out += "Definition: " + e.definition + "\n\n";
            out += "Synonyms: " + e.synonyms.join(", ") + "\n";
            out += "Antonyms: " + e.antonyms.join(", ") + "\n\n";
            out += "Background: " + e.background + "\n\n";
            out += "Usage: " + e.usage + "\n";
            resultOutputSearch->setPlainText(out);
            return;
        }
    }
    resultOutputSearch->setPlainText(tr("Not found"));
}

// Updates the browse list when a new letter is selected in the combo box.
void Gui_Holder::on_letterComboBox_currentIndexChanged(int index)
{
    if (index < 0) return;
    QChar letter = letterComboBox->itemText(index).at(0);
    // Get all words starting with that letter.
    auto list = WordStorage::instance().wordsForLetter(letter);
    browseOutput->clear();
    for (const auto &e : list) {
        QListWidgetItem *it = new QListWidgetItem(e.word + " - " + e.definition, browseOutput);
        it->setData(Qt::UserRole, e.word);
        browseOutput->addItem(it);
    }
}

// Slot: open WordDetailWindow when a browse item is clicked
void Gui_Holder::on_browseItem_clicked(QListWidgetItem *item)
{
    if (!item) return;
    QString key = item->data(Qt::UserRole).toString();
    if (key.isEmpty()) return;

    // Find the full WordEntry for the selected word
    auto all = WordStorage::instance().allWords();
    for (const auto &e : all) {
        if (e.word.compare(key, Qt::CaseInsensitive) == 0) {
            WordDetailWindow dlg(e, this);
            dlg.exec();
            return;
        }
    }
}

// Displays the current user's profile details in a new modal window.
void Gui_Holder::on_profileButton_clicked()
{
    QString username = UserStorage::instance().currentUser();
    if (username.isEmpty()) {
        QMessageBox::information(this, tr("Profile"), tr("No user is currently selected. Please add a
user first."));
        return;
    }

    // Force the UserStorage cache to reload data from the user's file.
    UserStorage::instance().loadUserData(username);

    // Retrieve the detailed user data and deserialize it into a User struct.
    QJsonObject data = UserStorage::instance().currentUserData();
    User u = User::fromJson(data);

    // Launch the new modal profile window.
    UserProfileWindow profileDlg(u, this);
    profileDlg.exec();
}

// Handles closing event: confirms exit and saves data.
void Gui_Holder::closeEvent(QCloseEvent *event)
{
    auto res = QMessageBox::question(this,
                                     tr("Confirm exit"),
                                     tr("Are you sure you want to exit the application?"),
                                     QMessageBox::Yes | QMessageBox::No,
                                     QMessageBox::No);
    if (res == QMessageBox::Yes) {
        // Ensure words are persisted to file storage.
        WordStorage::instance().save();
        event->accept();
    } else {
        event->ignore();
    }
}
```

Gui_Holder.cpp

```
#ifndef GUI_HOLDER_H
#define GUI_HOLDER_H

#include <QMainWindow>
#include <QLabel>
#include <QPixmap>
#include <QTabWidget>
#include <QLineEdit>
#include <QTextEdit>
#include <QComboBox>
#include <QListWidget>
#include <QPushButton>
#include <QCloseEvent>

// Forward Declarations
class Function;
class AboutWindow;

//The main window class for the DeepLingo application.
//Manages the layout, main tabs, and top-bar actions (Profile and Settings).
class Gui_Holder : public QMainWindow
```

```
{
    Q_OBJECT

public:
    explicit Gui_Holder(QWidget *parent = nullptr);
    ~Gui_Holder();

private slots:
    // Slots for main application buttons/interactions
    void on_profileButton_clicked();
    void on_settingsButton_clicked();
    void on_addWordButton_clicked();
    void on_searchWordButton_clicked();
    void on_letterComboBox_currentIndexChanged(int index);
    void on_browseItem_clicked(QListWidgetItem *item);

private:
    // UI Setup & Maintenance
    void setupUI();
    void updateProfileView();
    void updateProfileAvatar();
    void closeEvent(QCloseEvent *event) override;

    // Top Bar Widgets
    QLabel *m_logoLabel;
    QPixmap m_logoImage;
    QPushButton *profileButton;
    QPushButton *settingsButton;

    // Add Word Tab Widgets
    QLineEdit *wordInputAdd;
    QTextEdit *definitionInputAdd;
    QLineEdit *synonymsInput;
    QLineEdit *antonymsInput;
    QTextEdit *backgroundInput;
    QTextEdit *usageInput;
    QPushButton *addWordButton;

    // Search Tab Widgets
    QLineEdit *wordInputSearch;
    QPushButton *searchWordButton;
    QTextEdit *resultOutputSearch;

    // Browse Tab Widgets
    QComboBox *letterComboBox;
    QListWidget *browseOutput;
    // Application Functionality Holder
    Function *m_appFunction;
};

#endif // GUI_HOLDER_H
```

Gui_Holder.h

```
#include "GUI/LoadingScreen.h"
#include "Qt_includes.h"

LoadingScreen::LoadingScreen(QWidget *parent) :
    QDialog(parent, Qt::SplashScreen | Qt::FramelessWindowHint), // Frameless and Splash
Screen
    m_timer(new QTimer(this)),
    m_duration(0),
    m_elapsed(0)
{
    // Load the splash image
    if (m_splashImage.load("deeplingo_logo_loading.png")) {
        setFixedSize(m_splashImage.size());

        int imgWidth = m_splashImage.width();
        int imgHeight = m_splashImage.height();

        // --- Calculate m_progressBarRect based on proportions ---
        // These proportions are based on the provided logo image:
        int barWidth = static_cast<int>(imgWidth * 0.40);
        int barHeight = static_cast<int>(imgHeight * 0.06);
        int barX = (imgWidth - barWidth) / 2;
        int barY = static_cast<int>(imgHeight * 0.52);

        m_progressBarRect = QRect(barX, barY, barWidth, barHeight);
    } else {
        // Fallback if image not loaded
        setFixedSize(600, 400);
        m_progressBarRect = QRect(150, 300, 300, 30); // Default placeholder
        qWarning("LoadingScreen: Splash image 'deeplingo_logo_loading.png' not found or failed
to load.");
    }

    // --- Transparency ---
    setAttribute(Qt::WA_TranslucentBackground);
```

```cpp
    setWindowFlags(Qt::SplashScreen | Qt::FramelessWindowHint | Qt::WindowStaysOnTopHint |
Qt::NoDropShadowWindowHint);

    // Connect timer for progress updates
    connect(m_timer, &QTimer::timeout, this, &LoadingScreen::updateProgress);

    // Center the splash screen
    const QScreen *screen = QGuiApplication::primaryScreen();
    if (screen) {
        move(screen->geometry().center() - frameGeometry().center());
    }
}

void LoadingScreen::start(int durationMs) {
    m_duration = durationMs;
    m_elapsed = 0;
    show();
    m_timer->start(20); // Update progress every 20ms
}

void LoadingScreen::paintEvent(QPaintEvent *event) {
    Q_UNUSED(event);
    QPainter painter(this);
    painter.setRenderHint(QPainter::Antialiasing);

    // Draw the splash image
    if (!m_splashImage.isNull()) {
        painter.drawPixmap(0, 0, m_splashImage);
    } else {
        // Fallback: draw a simple box if the image is missing
        painter.fillRect(rect(), Qt::lightGray);
    }

    // Draw the custom progress bar
    if (m_duration > 0 && !m_splashImage.isNull()) {
        double progress = static_cast<double>(m_elapsed) / m_duration;
        int barWidth = static_cast<int>(m_progressBarRect.width() * progress);

        // 1. Draw the background/outline of the progress bar (the empty square)
        painter.setPen(QPen(QColor("#4CAF50"), 4));
        painter.setBrush(Qt::NoBrush);
        painter.drawRect(m_progressBarRect);

        // 2. Draw the filled part of the progress bar
        painter.setPen(Qt::NoPen);
        painter.setBrush(QColor("#4CAF50")); // Green fill
        // Create a slightly inset rectangle for the fill to look clean inside the border
        QRect fillRect = m_progressBarRect.adjusted(3, 3, -3, -3); // Inset by 3 pixels
        fillRect.setWidth(barWidth - 6); // Adjust width for the inset

        painter.drawRect(fillRect);
    }
}

void LoadingScreen::updateProgress() {
    m_elapsed += m_timer->interval();
    if (m_elapsed >= m_duration) {
        m_timer->stop();
        emit finished();
        close();
    }
    update(); // Request a repaint to update the progress bar
}
```

LoadingScreen.cpp

```
#ifndef LOADINGSCREEN_H
#define LOADINGSCREEN_H

#include <QDialog>
#include <QTimer>
#include <QPixmap>
#include <QRect>

class LoadingScreen : public QDialog {
    Q_OBJECT
public:
    explicit LoadingScreen(QWidget *parent = nullptr);
    void start(int durationMs);

signals:
    void finished(); // Emitted when the loading duration is over

protected:
    void paintEvent(QPaintEvent *event) override; // To draw the background image and progress
bar

private slots:
    void updateProgress();
private:
    QTimer *m_timer;
    int m_duration; // Total duration of the loading screen
    int m_elapsed;  // Elapsed time
    QPixmap m_splashImage; // The image for the splash screen

    // Position and size for the progress bar rectangle
    QRect m_progressBarRect;
};

#endif // LOADINGSCREEN_H
```

LoadingScreen.h

```
#include "GUI/UserProfileWindow.h"
#include "User_Files/UserStorage.h"
#include "User_Files/User.h"
#include "GUI/WordDetailWindow.h"
#include "Word_Files/Word_Storage.h"
#include "Qt_includes.h"


// --- CalendarAndNotesWidget Implementation ---
// Constructor: Initializes the calendar and notes widget layout and connections.
CalendarAndNotesWidget::CalendarAndNotesWidget(QWidget *parent) : QWidget(parent) {
    QVBoxLayout *mainLayout = new QVBoxLayout(this);
    mainLayout->setContentsMargins(0, 0, 0, 0);

    // 1. Calendar Button and Calendar Widget.
    m_calendarButton = new QPushButton(tr("Open Calendar"));

    m_calendarButton->setIcon(QApplication::style()->standardIcon(QStyle::SP_DirIcon));
    m_calendarButton->setIconSize(QSize(24, 24));
    m_calendarButton->setStyleSheet("border: none; background-color: #f0f2f5; color: #4CAF50;
font-weight: bold; text-align: left; padding: 5px;");

    m_calendar = new QCalendarWidget(this);
    m_calendar->setMaximumHeight(200);

    // Styling for QSpinBox/text-field to ensure month/year visibility.
    m_calendar->setMinimumSize(250, 180);
    // Calendar styles tuned to the app theme:
    // - navigation bar uses app green (#4CAF50)
```

```
   // - nav text (month/year) and controls are white for contrast
   // - day numbers use a dark color for readability
   // - selection highlight uses the app green with white text
   m_calendar->setStyleSheet(
       "QCalendarWidget QWidget#qt_calendar_navigationbar {"
       "    background-color: #4CAF50;"
       "    border-top-left-radius: 5px;"
       "    border-top-right-radius: 5px;"
       "    height: 30px;"
       "    color: white;"
       "}"
       "QCalendarWidget QWidget#qt_calendar_navigationbar QToolButton, QCalendarWidget
QWidget#qt_calendar_navigationbar QSpinBox, QCalendarWidget
QWidget#qt_calendar_navigationbar QSpinBox * {"
       "    color: white;"
       "    background-color: transparent;"
       "}"
       "QCalendarWidget QToolButton:hover {"
       "    background-color: rgba(56,142,60,0.85);" /* subtle darker green */
       "}"
       "QCalendarWidget QSpinBox::text-field {"
       "    color: white;"
       "    background-color: transparent;"
       "    border: none;"
       "    margin: 0px;"
       "    padding: 0px;"
       "}"
       "QCalendarWidget QSpinBox::up-button, QCalendarWidget QSpinBox::down-button {"
       "    subcontrol-origin: border;"
       "}"
       "QCalendarWidget QMenu {"
       "    background-color: white;"
       "}"
       "QCalendarWidget QAbstractItemView {"
       "    color: #333333;"
       "    background-color: transparent;"
       "    selection-background-color: #4CAF50;"
       "    selection-color: white;"
       "}"
       "QCalendarWidget QAbstractItemView::item:hover {"
       "    background-color: rgba(76,175,80,0.12);"
       "}"
       "QCalendarWidget QAbstractItemView::item:selected {"
       "    background-color: #4CAF50;"
       "    color: white;"
       "}"
```

```
        "QCalendarWidget QTableView {"
        "    gridline-color: rgba(0,0,0,0.04);"
        "}"
    );

    m_calendar->hide();

    // Container for the calendar and its button
    QVBoxLayout *calendarLayout = new QVBoxLayout();
    calendarLayout->addWidget(m_calendarButton);
    calendarLayout->addWidget(m_calendar);
    calendarLayout->setContentsMargins(0, 0, 0, 0);

    mainLayout->addLayout(calendarLayout); // Add the calendar container

    // 2. Notes Section.
    QLabel *notesTitle = new QLabel(tr("Notes:"));
    notesTitle->setStyleSheet("font-weight: bold; margin-top: 10px; color: #4CAF50;");

    m_notesEdit = new QTextEdit(this);
    m_notesEdit->setPlaceholderText(tr("Type notes here..."));
    m_notesEdit->setFixedHeight(150);

    m_notesSaveButton = new QPushButton(tr("Save Notes"));
    m_notesSaveButton->setStyleSheet("background-color: #4CAF50; color: white; border-radius:
5px; padding: 5px;");

    mainLayout->addWidget(notesTitle);
    mainLayout->addWidget(m_notesEdit);
    mainLayout->addWidget(m_notesSaveButton);
    mainLayout->addStretch();

    connect(m_calendarButton, &QPushButton::clicked, this,
&CalendarAndNotesWidget::on_calendarButton_clicked);
    connect(m_notesSaveButton, &QPushButton::clicked, this,
&CalendarAndNotesWidget::on_notesSaveButton_clicked);

    loadNotes();
}

//  Toggles the visibility of the calendar widget.
void CalendarAndNotesWidget::on_calendarButton_clicked() {
    m_calendar->setVisible(!m_calendar->isVisible());
    m_calendarButton->setText(m_calendar->isVisible() ? tr("Close Calendar") : tr("Open
Calendar"));
}
```

```
// Slot: Saves the notes content to user storage.
void CalendarAndNotesWidget::on_notesSaveButton_clicked() {
    saveNotes();
    QMessageBox::information(this, tr("Notes Saved"), tr("Notes content has been saved."));
}

// Loads the stored notes content for the current user into the text editor.
void CalendarAndNotesWidget::loadNotes() {
    QString username = UserStorage::instance().currentUser();
    if (username.isEmpty()) return;

    QJsonObject data = UserStorage::instance().currentUserData();
    QString notes = data.value("word_notes").toString();
    m_notesEdit->setText(notes);
}

// Saves the current text editor content back to the user's data storage.
void CalendarAndNotesWidget::saveNotes() {
    QString username = UserStorage::instance().currentUser();
    if (username.isEmpty()) return;

    QJsonObject data = UserStorage::instance().currentUserData();
    data["word_notes"] = m_notesEdit->toPlainText();
    UserStorage::instance().updateCurrentUserData(data);

    UserStorage::instance().saveCurrentUserData();
}


// --- UserProfileWindow Implementation ---
// Constructor: Sets window properties and calls setupUI.
UserProfileWindow::UserProfileWindow(const User &user, QWidget *parent)
    : QDialog(parent)
{
    setWindowTitle(tr("Profile: %1").arg(user.name));
    setMinimumSize(600, 450);
    setStyleSheet("background-color: #ffffff;");

    setupUI(user);
}

// Sets up the main layout and widgets for the profile window.
void UserProfileWindow::setupUI(const User &user) {
    QHBoxLayout *mainLayout = new QHBoxLayout(this);
```

```
   // --- Left Pane: User Details ---
   QWidget *detailsPane = new QWidget();
   QVBoxLayout *vLayout = new QVBoxLayout(detailsPane);
   vLayout->setAlignment(Qt::AlignTop);

   QLabel *title = new QLabel(tr("User Details"));
   QFont titleFont = title->font();
   titleFont.setPointSize(16);
   titleFont.setBold(true);
   title->setFont(titleFont);
   title->setStyleSheet("color: #4CAF50; margin-bottom: 15px;");
   vLayout->addWidget(title);

   m_nameLabel = new QLabel(tr("Name: <b>%1</b>").arg(user.name));
   m_ageLabel = new QLabel(tr("Age: <b>%1</b>").arg(user.age));
   vLayout->addWidget(m_nameLabel);
   vLayout->addWidget(m_ageLabel);
   vLayout->addSpacing(20);

   // Added Words section.
   m_addedWordsLabel = new QLabel(tr("<b>Words Added
(%1):</b>").arg(user.addedWords.size()));

   // Use QListWidget for interactable words.
   m_addedWordsList = new QListWidget();
   m_addedWordsList->setSelectionMode(QAbstractItemView::SingleSelection);
   m_addedWordsList->setFixedHeight(100);
   m_addedWordsList->setStyleSheet("border: 1px solid #d0d0d0; border-radius: 5px;");

   if (user.addedWords.isEmpty()) {
      QListWidgetItem *placeholderItem = new QListWidgetItem(tr("None recorded."),
m_addedWordsList);
      placeholderItem->setFlags(placeholderItem->flags() & ~Qt::ItemIsSelectable); // Disable
selection
   } else {
      // Add words to the list, ensuring uniqueness.
      QStringList uniqueWords = user.addedWords;
      uniqueWords.removeDuplicates();
      m_addedWordsList->addItems(uniqueWords);
   }

   // Connect list click signal to the slot.
   connect(m_addedWordsList, &QListWidget::itemClicked, this,
&UserProfileWindow::on_addedWordClicked);
```

```
    // Recent Searches section.
    QLabel *recentSearchesTitle = new QLabel(tr("<b>Recent Searches
(%1):</b>").arg(user.recentSearches.size()));
    m_recentSearchesText = new QTextEdit();
    m_recentSearchesText->setReadOnly(true);
    m_recentSearchesText->setFixedHeight(120);
    m_recentSearchesText->setStyleSheet("background-color: #f8f8f8; border: 1px solid #d0d0d0;
border-radius: 5px;");

    QString searchesText = user.recentSearches.isEmpty() ? tr("No recent searches recorded.") :
user.recentSearches.join("\n");
    m_recentSearchesText->setText(searchesText);

    vLayout->addWidget(m_addedWordsLabel);
    vLayout->addWidget(m_addedWordsList);
    vLayout->addSpacing(15);
    vLayout->addWidget(recentSearchesTitle);
    vLayout->addWidget(m_recentSearchesText);
    vLayout->addStretch(1);

    detailsPane->setStyleSheet("padding-right: 20px; border-right: 1px solid #e0e0e0;");


    // --- Right Pane: Calendar and Notes ---
    CalendarAndNotesWidget *sidebar = new CalendarAndNotesWidget();
    sidebar->setFixedWidth(250);

    // Add both panes to the main layout.
    mainLayout->addWidget(detailsPane, 2);
    mainLayout->addWidget(sidebar, 1);
}

// Slot: Handles clicks on words in the list to open the detail window.
void UserProfileWindow::on_addedWordClicked(QListWidgetItem *item) {
    QString wordKey = item->text();

    // Prevent clicking the placeholder item.
    if (wordKey == tr("None recorded.")) return;

    // Look up the word details in WordStorage.
    auto words = WordStorage::instance().allWords();
    for (const auto &e : words) {
        if (e.word.compare(wordKey, Qt::CaseInsensitive) == 0) {
            // Found the word, open the detail window using the WordEntry struct.
            WordDetailWindow detailDlg(e, this);
            detailDlg.exec();
```

```
         return;
      }
   }
   QMessageBox::warning(this, tr("Error"), tr("Word details not found in storage."));
}
```

UserProfileWindow.cpp

```
#ifndef USERPROFILEWINDOW_H
#define USERPROFILEWINDOW_H

#include <QDialog>
#include <QLabel>
#include <QTextEdit>
#include <QPushButton>
#include <QListWidget>
#include <QListWidgetItem>
#include <QHBoxLayout>
#include <QVBoxLayout>

// Forward declarations
class QCalendarWidget;
class QSplitter;
struct User;

// Widget containing the Calendar and Notes sections for the sidebar.
class CalendarAndNotesWidget : public QWidget {
   Q_OBJECT
public:
   CalendarAndNotesWidget(QWidget *parent = nullptr);

private slots:
   void on_calendarButton_clicked();
   void on_notesSaveButton_clicked();

private:
   QPushButton *m_calendarButton;
   QCalendarWidget *m_calendar;
   QTextEdit *m_notesEdit;
   QPushButton *m_notesSaveButton;

   // Loads notes from the user data storage.
   void loadNotes();
   // Saves the current text editor content to user data storage.
```

```
    void saveNotes();
};


// Main dialog window for displaying the User Profile details.
class UserProfileWindow : public QDialog {
    Q_OBJECT
public:
    UserProfileWindow(const User &user, QWidget *parent = nullptr);

private slots:
    // Slot to handle clicks on added words in the list.
    void on_addedWordClicked(QListWidgetItem *item);

private:
    QLabel *m_nameLabel;
    QLabel *m_ageLabel;
    QLabel *m_addedWordsLabel;
    QListWidget *m_addedWordsList;
    QTextEdit *m_recentSearchesText;

    // Sets up the user interface elements based on the provided User struct.
    void setupUI(const User &user);
};

#endif // USERPROFILEWINDOW_H
```

UserProfileWindow.h

```
#include "GUI/WordDetailWindow.h"
#include "Qt_includes.h"

// Constructor: Initializes the dialog window and sets up the UI.
WordDetailWindow::WordDetailWindow(const WordEntry &wordData, QWidget *parent)
    : QDialog(parent)
{
    setWindowTitle(tr("Word Details: %1").arg(wordData.word));
    setMinimumSize(450, 600);
    setStyleSheet("background-color: #f8f8f8;");

    setupUI(wordData);
}

// Sets up the UI elements based on the provided word data.
```

```
void WordDetailWindow::setupUI(const WordEntry &wordData)
{
    QVBoxLayout *mainLayout = new QVBoxLayout(this);

    QScrollArea *scrollArea = new QScrollArea(this);
    scrollArea->setWidgetResizable(true);
    scrollArea->setFrameShape(QFrame::NoFrame);

    QWidget *contentWidget = new QWidget();
    QVBoxLayout *contentLayout = new QVBoxLayout(contentWidget);
    contentLayout->setContentsMargins(20, 20, 20, 20);
    contentLayout->setSpacing(15);
    contentLayout->setAlignment(Qt::AlignTop);

    // Word Title
    wordTitle = new QLabel(wordData.word);
    QFont titleFont = wordTitle->font();
    titleFont.setPointSize(28);
    titleFont.setBold(true);
    wordTitle->setFont(titleFont);
    wordTitle->setStyleSheet("color: #4CAF50;");
    contentLayout->addWidget(wordTitle);

    // Definition (with small Tagalog toggle button on the right)
    QLabel *defLabel = new QLabel(tr("<b>Definition:</b>"));
    // Row for the label and the small toggle button
    QHBoxLayout *defRow = new QHBoxLayout();
    defRow->addWidget(defLabel);
    defRow->addStretch();

    toggleLangBtn = new QPushButton(this);
    toggleLangBtn->setToolTip(tr("Toggle Tagalog translation"));
    // Use embedded SVG icon for clearer UX; fall back to text if icon missing
    QIcon transIcon(QStringLiteral(":/icons/translate.svg"));
    if (!transIcon.isNull()) {
        toggleLangBtn->setIcon(transIcon);
        toggleLangBtn->setIconSize(QSize(18,18));
        toggleLangBtn->setText("");
    } else {
        toggleLangBtn->setText("TL");
    }
    // Make the button more visible: larger circular green button with white icon
    toggleLangBtn->setFixedSize(36, 36);
    toggleLangBtn->setFlat(true);
    toggleLangBtn->setStyleSheet(
        "QPushButton { background-color: #4CAF50; color: white; border-radius: 18px; }"
        "QPushButton:hover { background-color: #45a049; }"
    );
    defRow->addWidget(toggleLangBtn);
    contentLayout->addLayout(defRow);

    defText = new QTextEdit();
    defText->setPlainText(wordData.definition);
    defText->setReadOnly(true);
    defText->setFixedHeight(80);
    defText->setStyleSheet("background-color: white; border: 1px solid #d0d0d0; border-radius:
5px; padding: 5px;");
    contentLayout->addWidget(defText);

    // Synonyms
    QLabel *synLabel = new QLabel(tr("<b>Synonyms:</b>"));
    synText = new QLabel(wordData.synonyms.join(", "));
    synText->setWordWrap(true);
    contentLayout->addWidget(synLabel);
    contentLayout->addWidget(synText);

    // Antonyms
    QLabel *antLabel = new QLabel(tr("<b>Antonyms:</b>"));
    antText = new QLabel(wordData.antonyms.join(", "));
    antText->setWordWrap(true);
    contentLayout->addWidget(antLabel);
    contentLayout->addWidget(antText);

    // Background / Etymology
    QLabel *bgLabel = new QLabel(tr("<b>Background / Etymology:</b>"));
    bgText = new QTextEdit();
    bgText->setPlainText(wordData.background);
    bgText->setReadOnly(true);
    bgText->setFixedHeight(100);
    bgText->setStyleSheet("background-color: white; border: 1px solid #d0d0d0; border-radius:
5px; padding: 5px;");
    contentLayout->addWidget(bgLabel);
    contentLayout->addWidget(bgText);

    // Usage / Example
    QLabel *usageLabel = new QLabel(tr("<b>Usage / Example:</b>"));
    usageText = new QTextEdit();
    usageText->setPlainText(wordData.usage);
    usageText->setReadOnly(true);
    usageText->setFixedHeight(80);
    usageText->setStyleSheet("background-color: white; border: 1px solid #d0d0d0; border-radius:
5px; padding: 5px;");
```

```
    contentLayout->addWidget(usageLabel);
    contentLayout->addWidget(usageText);

    contentLayout->addStretch(1);
    scrollArea->setWidget(contentWidget);
    mainLayout->addWidget(scrollArea);

    // Connect toggle button signal to switch between English and Tagalog
    // Prepare English originals so we can restore them when toggling back.
    m_engDef = wordData.definition;
    m_engSyn = wordData.synonyms.join(", ");
    m_engAnt = wordData.antonyms.join(", ");
    m_engBg = wordData.background;
    m_engUsage = wordData.usage;
    m_engWord = wordData.word;

    // Parse labeled Tagalog sections from the single `translation` field when possible.
    // Expected markers (in Tagalog data): "Kasingkahulugan:", "Kasalungat:", "Halimbawa:".
    auto parseTranslation = [](const QString &t) {
        QString defOut;
        QString synOut;
        QString antOut;
        QString bgOut;
        QString usageOut;

        if (t.isEmpty()) return
std::tuple<QString,QString,QString,QString,QString>(defOut,synOut,antOut,bgOut,usageOut);

        // Split lines and look for markers
        QStringList lines = t.split('\n', Qt::SkipEmptyParts);
        QStringList pending;
        for (const QString &ln : lines) {
            QString L = ln.trimmed();
            if (L.startsWith("Kasingkahulugan", Qt::CaseInsensitive)) {
                // remove marker and collect
                QString v = L.section(':', 1).trimmed();
                synOut = v;
            } else if (L.startsWith("Kasalungat", Qt::CaseInsensitive)) {
                QString v = L.section(':', 1).trimmed();
                antOut = v;
            } else if (L.startsWith("Halimbawa", Qt::CaseInsensitive)) {
                QString v = L.section(':', 1).trimmed();
                usageOut = v;
            } else {
                pending.append(L);
            }
        }

        // If pending lines exist, treat the first as the Tagalog definition
        if (!pending.isEmpty()) {
            defOut = pending.takeFirst();
            // any remaining pending lines append to background if background empty
            if (!pending.isEmpty()) bgOut = pending.join("\n");
        }

        return
std::tuple<QString,QString,QString,QString,QString>(defOut,synOut,antOut,bgOut,usageOut);
    };

    const QString tagall = wordData.translation;
    std::tie(m_tagDef, m_tagSyn, m_tagAnt, m_tagBg, m_tagUsage) = parseTranslation(tagall);
    // tagDef is typically the first pending line; use it as Tagalog title when present
    m_tagTitle = m_tagDef;

    connect(toggleLangBtn, &QPushButton::clicked, this, [this]() {
        static bool showingTagalog = false;
        if (showingTagalog) {
            defText->setPlainText(m_engDef);
            synText->setText(m_engSyn);
            antText->setText(m_engAnt);
            bgText->setPlainText(m_engBg);
            usageText->setPlainText(m_engUsage);
            // restore English title
            wordTitle->setText(m_engWord);
            setWindowTitle(tr("Word Details: %1").arg(m_engWord));
            toggleLangBtn->setToolTip(tr("Show Tagalog translation"));
            // If icon fallback text is used, show TL
            if (toggleLangBtn->text().isEmpty()) {
                // keep icon only
            } else {
                toggleLangBtn->setText("TL");
            }
        } else {
            // If there is no structured tagalog content, show message and return
            if (m_tagDef.isEmpty() && m_tagSyn.isEmpty() && m_tagAnt.isEmpty() &&
m_tagBg.isEmpty() && m_tagUsage.isEmpty()) {
                QMessageBox::information(this, tr("No translation"), tr("No Tagalog translation
available for this word."));
                return;
            }
            // use Tagalog parts if present, otherwise keep English
            defText->setPlainText(!m_tagDef.isEmpty() ? m_tagDef : m_engDef);
            synText->setText(!m_tagSyn.isEmpty() ? m_tagSyn : m_engSyn);
            antText->setText(!m_tagAnt.isEmpty() ? m_tagAnt : m_engAnt);
            bgText->setPlainText(!m_tagBg.isEmpty() ? m_tagBg : m_engBg);
            usageText->setPlainText(!m_tagUsage.isEmpty() ? m_tagUsage : m_engUsage);
            // set Tagalog title if available
            if (!m_tagTitle.isEmpty()) {
                wordTitle->setText(m_tagTitle);
                setWindowTitle(tr("Word Details: %1").arg(m_tagTitle));
            }
            toggleLangBtn->setToolTip(tr("Show English definition"));
            if (toggleLangBtn->text().isEmpty()) {
                // keep icon only
            } else {
                toggleLangBtn->setText("EN");
            }
        }
        showingTagalog = !showingTagalog;
    });
}
```

WordDetailWindow.cpp

```
#ifndef WORDDETAILWINDOW_H
#define WORDDETAILWINDOW_H

#include <QDialog>
#include <QObject>
#include <QLabel>
#include <QTextEdit>
#include <QPushButton>

// Forward declaration
struct WordEntry;

// Dialog window to display the full details of a specific word entry.
class WordDetailWindow : public QDialog {
    Q_OBJECT
public:
    // Constructor: Takes a constant reference to a WordEntry struct.
    explicit WordDetailWindow(const WordEntry &wordData, QWidget *parent = nullptr);

private:
    // Sets up the UI elements based on the provided WordEntry data.
    void setupUI(const WordEntry &wordData);
    // Member variables for toggling translation
    // Section widgets that may be swapped to Tagalog equivalents
    QLabel *wordTitle;
    QTextEdit *defText;
    QLabel *synText;
    QLabel *antText;
    QTextEdit *bgText;
    QTextEdit *usageText;
    QPushButton *toggleLangBtn;

    // Stored originals and parsed translations to avoid capturing local variables
    QString m_engWord;
    QString m_engDef;
    QString m_engSyn;
    QString m_engAnt;
    QString m_engBg;
    QString m_engUsage;

    QString m_tagDef;
    QString m_tagSyn;
    QString m_tagAnt;
    QString m_tagBg;
    QString m_tagUsage;
    QString m_tagTitle;
};

#endif // WORDDETAILWINDOW_H
```

WordDetailWindow.h

```
#ifndef USER_H
#define USER_H

#include <QString>
#include <QJsonObject>
#include <QJsonArray>
#include <QStringList>

// Simple POD for a user record (name + age) with JSON serialization helpers.
struct User {
    QString name;           // user's name
    int age = 0;            // user's age
    QStringList addedWords;    // words this user added
```

```
   QStringList recentSearches; // recent search terms, most-recent-first

   // Convert this user to a QJsonObject for saving.
   QJsonObject toJson() const {
      QJsonObject obj;
      obj["name"] = name;
      obj["age"] = age;

      QJsonArray addedArr;
      for (const QString &w : addedWords) addedArr.append(w);
      obj["addedWords"] = addedArr;

      QJsonArray recentArr;
      for (const QString &s : recentSearches) recentArr.append(s);
      obj["recentSearches"] = recentArr;

      return obj;
   }

   // Construct a User from a QJsonObject loaded from disk.
   static User fromJson(const QJsonObject &obj) {
      User u;
      u.name = obj.value("name").toString();
      u.age = obj.value("age").toInt(0);

      u.addedWords.clear();
      QJsonValue addedVal = obj.value("addedWords");
      if (addedVal.isArray()) {
         QJsonArray addedArr = addedVal.toArray();
         for (const QJsonValue &v : addedArr) if (v.isString()) u.addedWords.append(v.toString());
      }

      u.recentSearches.clear();
      QJsonValue recentVal = obj.value("recentSearches");
      if (recentVal.isArray()) {
         QJsonArray recentArr = recentVal.toArray();
         for (const QJsonValue &v : recentArr) if (v.isString())
u.recentSearches.append(v.toString());
      }

      return u;
   }
};

#endif // USER_H
```

User.h

```
#include "User_Files/UserDialog.h"
#include "User_Files/UserStorage.h"
#include "User_Files/User.h"
#include "Qt_includes.h"

// Construct the dialog UI (name + age + OK/Cancel).
UserDialog::UserDialog(QWidget *parent) : QDialog(parent) {
   setWindowTitle("Enter User Info");
   setModal(true);

   m_nameEdit = new QLineEdit(this);
   m_ageSpin = new QSpinBox(this);
   m_ageSpin->setRange(0, 150); // reasonable age range

   m_okBtn = new QPushButton("OK", this);
   m_cancelBtn = new QPushButton("Cancel", this);

   QHBoxLayout *nameRow = new QHBoxLayout();
   nameRow->addWidget(new QLabel("Name:"));
   nameRow->addWidget(m_nameEdit);

   QHBoxLayout *ageRow = new QHBoxLayout();
   ageRow->addWidget(new QLabel("Age:"));
   ageRow->addWidget(m_ageSpin);

   QHBoxLayout *buttons = new QHBoxLayout();
   buttons->addStretch();
   buttons->addWidget(m_okBtn);
   buttons->addWidget(m_cancelBtn);

   QVBoxLayout *main = new QVBoxLayout(this);
   main->addLayout(nameRow);
   main->addLayout(ageRow);
   main->addLayout(buttons);

   connect(m_okBtn, &QPushButton::clicked, this, &UserDialog::onAccept);
   connect(m_cancelBtn, &QPushButton::clicked, this, &UserDialog::reject);
}

// Validate inputs, add user to storage, then close with accept().
void UserDialog::onAccept() {
   QString name = m_nameEdit->text().trimmed();
   int age = m_ageSpin->value();
```

```
    if (name.isEmpty()) {
        QMessageBox::warning(this, "Validation", "Please enter a name.");
        return; // don't close
    }

    User u;
    u.name = name;
    u.age = age;

    UserStorage::instance().addUser(u); // store user

    // Set the newly created user as the current user.
    UserStorage::instance().setCurrentUser(name);

    accept();
}
```

UserDialog.cpp

```
#ifndef USERDIALOG_H
#define USERDIALOG_H

#include <QDialog>

// Modal dialog that collects a user's name and age.
class QLineEdit;
class QSpinBox;
class QPushButton;

class UserDialog : public QDialog {
    Q_OBJECT
public:
    explicit UserDialog(QWidget *parent = nullptr);

private slots:
    // Called when the OK button is pressed; validates and saves the user.
    void onAccept();

private:
    QLineEdit *m_nameEdit;   // input for name
    QSpinBox *m_ageSpin;     // input for age
    QPushButton *m_okBtn;    // OK button
    QPushButton *m_cancelBtn;// Cancel button
};

#endif // USERDIALOG_H
```

UserDialog.h

```
#include "User_Files/UserStorage.h"
#include "User_Files/User.h"
#include "Qt_includes.h"

UserStorage &UserStorage::instance()
{
    static UserStorage s;
    return s;
}

// Loads the list of all users from the index file
bool UserStorage::load(const QString &indexPath)
{
    m_indexPath = indexPath;
    m_userIndex.clear();
    QFile f(m_indexPath);
    if (!f.exists()) {
        // ensure users dir exists and create empty index
        QDir().mkpath("users");
        return save(m_indexPath);
    }
    if (!f.open(QIODevice::ReadOnly | QIODevice::Text))
        return false;
    QJsonDocument doc = QJsonDocument::fromJson(f.readAll());
    f.close();
    if (!doc.isObject()) return false;
    QJsonObject root = doc.object();
    QJsonArray list = root.value("users").toArray();
    for (const auto &v : list) {
        if (!v.isObject()) continue;
        QJsonObject u = v.toObject();
        QString name = u.value("username").toString();
        if (name.isEmpty()) continue;
        m_userIndex.insert(name, u);
    }
    return true;
}
```

```
// Saves the list of all users to the index file
bool UserStorage::save(const QString &indexPath)
{
    QString path = indexPath.isEmpty() ? m_indexPath : indexPath;
    if (path.isEmpty()) return false;
    QJsonArray arr;
    for (auto it = m_userIndex.constBegin(); it != m_userIndex.constEnd(); ++it) {
        QJsonObject obj = it.value();
        if (obj.isEmpty()) obj["username"] = it.key();
        arr.append(obj);
    }
    QJsonObject root;
    root["users"] = arr;
    QJsonDocument doc(root);
    QFile f(path);
    QDir().mkpath(QFileInfo(path).absolutePath());
    if (!f.open(QIODevice::WriteOnly | QIODevice::Text)) return false;
    f.write(doc.toJson(QJsonDocument::Indented));
    f.close();
    QDir().mkpath("users");
    return true;
}

// Returns a list of all usernames
QStringList UserStorage::users() const
{
    return m_userIndex.keys();
}

// Checks if a user exists in the index
bool UserStorage::hasUser(const QString &username) const
{
    return m_userIndex.contains(username);
}

// Adds a new user by name only (minimal metadata)
bool UserStorage::addUser(const QString &username)
{
    if (username.isEmpty() || hasUser(username)) return false;
    QJsonObject meta;
    meta["username"] = username;
    m_userIndex.insert(username, meta);
    // create empty user data file
    QJsonObject data;
    data["created"] = true;
    m_userDataCache.insert(username, data);
    QDir().mkpath("users");
    saveUserData(username);
    save();
    return true;
}

// Added: Adds a new user based on the full User struct, and saves detailed data
bool UserStorage::addUser(const User &u)
{
    if (u.name.isEmpty() || hasUser(u.name)) return false;

    // 1. Update the main index with minimal metadata
    QJsonObject meta;
    meta["username"] = u.name;
    m_userIndex.insert(u.name, meta);

    // 2. Cache the detailed user data (User::toJson() handles age, lists, etc.)
    m_userDataCache.insert(u.name, u.toJson());

    // Ensure "users" directory exists and save the individual user file
    QDir().mkpath("users");
    saveUserData(u.name);

    // 3. Save the index file to persist the new user's existence
    save();
    return true;
}

// Removes a user and their data
bool UserStorage::removeUser(const QString &username)
{
    if (!hasUser(username)) return false;
    m_userIndex.remove(username);
    m_userDataCache.remove(username);
    save();
    QFile::remove(QString("users/%1.json").arg(username));
    if (m_currentUser == username) m_currentUser.clear();
    return true;
}

// Sets the current active user and loads their data
bool UserStorage::setCurrentUser(const QString &username)
{
    if (!hasUser(username)) return false;
    m_currentUser = username;
    return loadUserData(username);
```

```
}

// Returns the name of the currently active user
QString UserStorage::currentUser() const
{
    return m_currentUser;
}

// Loads detailed JSON data for a specific user
bool UserStorage::loadUserData(const QString &username)
{
    if (!hasUser(username)) return false;
    QString file = QString("users/%1.json").arg(username);
    QFile f(file);
    if (!f.exists()) {
        // create empty
        m_userDataCache.insert(username, QJsonObject());
        saveUserData(username);
        return true;
    }
    if (!f.open(QIODevice::ReadOnly | QIODevice::Text)) return false;
    QJsonDocument doc = QJsonDocument::fromJson(f.readAll());
    f.close();
    if (!doc.isObject()) return false;
    m_userDataCache.insert(username, doc.object());
    return true;
}

// Saves detailed JSON data for a specific user
bool UserStorage::saveUserData(const QString &username) const
{
    if (!m_userIndex.contains(username)) return false;
    QString file = QString("users/%1.json").arg(username);
    QJsonObject data = m_userDataCache.value(username, QJsonObject());
    QJsonDocument doc(data);
    QFile f(file);
    QDir().mkpath(QFileInfo(file).absolutePath());
    if (!f.open(QIODevice::WriteOnly | QIODevice::Text)) return false;
    f.write(doc.toJson(QJsonDocument::Indented));
    f.close();
    return true;
}

// Saves data for the currently active user
bool UserStorage::saveCurrentUserData() const
{
    if (m_currentUser.isEmpty()) return false;
    return saveUserData(m_currentUser);
}

// Returns the detailed JSON data for the current user
QJsonObject UserStorage::currentUserData() const
{
    return m_userDataCache.value(m_currentUser, QJsonObject());
}

// Updates the cached data for the current user
void UserStorage::updateCurrentUserData(const QJsonObject &data)
{
    if (m_currentUser.isEmpty()) return;
    m_userDataCache.insert(m_currentUser, data);
}
```

UserStorage.cpp

```
#ifndef USERSTORAGE_H
#define USERSTORAGE_H

#include <QString>
#include <QMap>
#include <QJsonObject>
#include <QStringList>
#include "User_Files/User.h"

class UserStorage {
public:
    static UserStorage &instance(); // Accesses the single instance of UserStorage

    // load / save index file (users.json)
    bool load(const QString &indexPath = QString("users.json")); // Loads the list of all users
    bool save(const QString &indexPath = QString()); // Saves the list of all users

    // index operations
    QStringList users() const; // Returns a list of all usernames
    bool hasUser(const QString &username) const; // Checks if a user exists in the index
    bool addUser(const QString &username); // Adds a new user by name only
    bool addUser(const User &user); // Added: Adds a new user based on the full User struct
    bool removeUser(const QString &username); // Removes a user and their data

    // current user
    bool setCurrentUser(const QString &username); // Sets the current active user and loads their
data
    QString currentUser() const; // Returns the name of the currently active user

    // per-user data
    bool loadUserData(const QString &username); // Loads detailed JSON data for a specific user
    bool saveUserData(const QString &username) const; // Saves detailed JSON data for a specific
user
    bool saveCurrentUserData() const; // Saves data for the currently active user

    QJsonObject currentUserData() const; // Returns the detailed JSON data for the current user
    void updateCurrentUserData(const QJsonObject &data); // Updates the cached data for the
current user

private:
    UserStorage() = default;
    QString m_indexPath;
    QMap<QString, QJsonObject> m_userIndex;     // metadata stored in index
    QMap<QString, QJsonObject> m_userDataCache; // actual per-user data
    QString m_currentUser;
};

#endif // USERSTORAGE_H
```

UserStorage.h

```
#include "Word_Files/Word_Storage.h"
#include "Qt_includes.h"

WordStorage &WordStorage::instance()
{
    static WordStorage s;
    return s;
}

bool WordStorage::load(const QString &path)
{
    m_path = path.isEmpty() ? QString("words.json") : path;
    m_words.clear();

    // Ensure initial in-code words are present first.
    insertInitialWords();

    QFile f(m_path);
    if (!f.exists()) {
        // No file yet: persist the initial in-code words to disk.
        QDir().mkpath(QFileInfo(m_path).absolutePath());
        save();
        return true;
    }

    if (!f.open(QIODevice::ReadOnly | QIODevice::Text)) return false;

    QJsonDocument doc = QJsonDocument::fromJson(f.readAll());
    f.close();

    if (!doc.isArray()) return false;

    // Load file entries but avoid duplicating words already provided by
    // insertInitialWords() (case-insensitive match by word text).
    QJsonArray arr = doc.array();
    for (const auto &v : arr) {
        if (!v.isObject()) continue;
        WordEntry entry = WordEntry::fromJson(v.toObject());

        bool exists = false;
        for (const auto &w : m_words) {
            if (w.word.compare(entry.word, Qt::CaseInsensitive) == 0) {
                exists = true;
                break;
            }
        }
        if (!exists) m_words.append(entry);
    }
    return true;
}

bool WordStorage::save(const QString &path)
{
    QString p = path.isEmpty() ? m_path : path;
    if (p.isEmpty()) return false;

    QJsonArray arr;
    for (const auto &w : m_words) arr.append(w.toJson());

    QJsonDocument doc(arr);
    QFile f(p);
    if (!f.open(QIODevice::WriteOnly | QIODevice::Text)) return false;
    f.write(doc.toJson(QJsonDocument::Indented));
    f.close();
    return true;
}

void WordStorage::addWord(const WordEntry &entry)
{
    for (const auto &w : m_words) {
        if (w.word.compare(entry.word, Qt::CaseInsensitive) == 0) return;
    }
    m_words.append(entry);
}

QVector<WordEntry> WordStorage::allWords() const { return m_words; }

QVector<WordEntry> WordStorage::wordsForLetter(QChar letter) const
{
    QVector<WordEntry> out;
    if (letter.isNull()) return out;

    QChar L = letter.toLower();
    for (const auto &w : m_words) {
        if (w.word.isEmpty()) continue;
        if (w.word.at(0).toLower() == L) out.append(w);
    }
    return out;
}

void WordStorage::insertInitialWords()
{
    m_words.clear();
    WordEntry e;

    // Letter A Words
    e.word = "Abandon";
    e.definition = "He had to abandon his plans when the storm hit.";
    e.translation = "Iwan\nKasingkahulugan: talikuran, pabayaan, isuko\nKasalungat: panatilihin,
ingatan, yakapin\nHalimbawa: Kinailangan niyang iwan ang kanyang mga plano nang dumating
ang bagyo.";
    e.synonyms = {"desert", "forsake", "relinquish"};
    e.antonyms = {"retain", "keep", "embrace"};
    e.background = "Origin: Old French 'abandoner', to surrender";
    e.usage = "He had to abandon his plans when the storm hit.";
    m_words.append(e);

    e.word = "Abolish";
    e.definition = "The law was abolished after it was deemed unjust.";
    e.translation = "Alisin (Pawiin)\nKasingkahulugan: burahin, alisin, kanselahin\nKasalungat:
itatag, likhain, panatilihin\nHalimbawa: Inalis ang batas matapos itong ideklarang hindi
makatarungan.";
    e.synonyms = {"eradicate", "eliminate", "cancel"};
    e.antonyms = {"establish", "create", "uphold"};
    e.background = "Origin: Latin 'abolere', to destroy";
    e.usage = "The law was abolished after it was deemed unjust.";
    m_words.append(e);

    e.word = "Absorb";
    e.definition = "The sponge will absorb all the water from the spill.";
    e.translation = "Sumipsip\nKasingkahulugan: sumipsip, tanggapin, lamunin\nKasalungat:
ilabas, pakawalan, palabasin\nHalimbawa: Sisisipsipin ng espongha ang lahat ng tubig sa
natapon.";
    e.synonyms = {"soak up", "take in", "assimilate"};
    e.antonyms = {"expel", "release", "discharge"};
    e.background = "Origin: Latin 'absorbere', to swallow up";
    e.usage = "The sponge will absorb all the water from the spill.";
    m_words.append(e);

    e.word = "Abundant";
    e.definition = "The region is abundant in natural resources.";
    e.translation = "Masagana\nKasingkahulugan: marami, sagana, labis\nKasalungat: kulang,
limitado, kapos\nHalimbawa: Sagana ang rehiyon sa likas na yaman.";
    e.synonyms = {"plentiful", "ample", "copious"};
    e.antonyms = {"scarce", "limited", "inadequate"};
    e.background = "Origin: Latin 'abundare', to overflow";
    e.usage = "The region is abundant in natural resources.";
    m_words.append(e);

    e.word = "Accelerate";
    e.definition = "The car started to accelerate as we drove downhill.";
    e.translation = "Pabilisin\nKasingkahulugan: pabilisin, paharapin, magmadali\nKasalungat:
pabagalin, huminto, ipagpaliban\nHalimbawa: Nagsimulang pabilisin ng kotse ang takbo pababa
ng burol.";
    e.synonyms = {"speed up", "hasten", "quicken"};
    e.antonyms = {"slow down", "decelerate", "delay"};
    e.background = "Origin: Latin 'accelerare', to hasten";
    e.usage = "The car started to accelerate as we drove downhill.";
    m_words.append(e);

    e.word = "Accessible";
    e.definition = "The library is accessible to everyone, including people with disabilities.";
    e.translation = "Madaling Mapuntahan\nKasingkahulugan: maabot, makuha,
magamit\nKasalungat: hindi maabot, sarado, harang\nHalimbawa: Madaling mapuntahan ng lahat
ang aklatan, kabilang ang may kapansanan.";
```

```
    e.synonyms = {"reachable", "attainable", "available"};
    e.antonyms = {"inaccessible", "unreachable", "blocked"};
    e.background = "Origin: Latin 'accessibilis', easy to reach";
    e.usage = "The library is accessible to everyone, including people with disabilities.";
    m_words.append(e);

    e.word = "Accomplish";
    e.definition = "She managed to accomplish all her goals for the year.";
    e.translation = "Matupad\nKasingkahulugan: makamit, matapos, maganap\nKasalungat:
mabigo, mapalampas, pabayaan\nHalimbawa: Nagawa niyang matupad ang lahat ng kanyang
mga layunin sa taon.";
    e.synonyms = {"achieve", "complete", "fulfill"};
    e.antonyms = {"fail", "miss", "neglect"};
    e.background = "Origin: Middle English 'accomplisshen', to achieve";
    e.usage = "She managed to accomplish all her goals for the year.";
    m_words.append(e);

    e.word = "Accurate";
    e.definition = "The test results were accurate and showed no errors.";
    e.translation = "Tumpak\nKasingkahulugan: tama, eksakto, wasto\nKasalungat: mali, hindi
tama, malabo\nHalimbawa: Tumpak ang mga resulta ng pagsusuri at walang mali.";
    e.synonyms = {"correct", "precise", "exact"};
    e.antonyms = {"inaccurate", "incorrect", "imprecise"};
    e.background = "Origin: Latin 'accuratus', well done";
    e.usage = "The test results were accurate and showed no errors.";
    m_words.append(e);

    // Letter B Words
    e.word = "Balance";
    e.definition = "She struggled to find a balance between work and personal life.";
    e.translation = "Balanse\nKasingkahulugan: pagkakapantay, katatagan, simetriya\nKasalungat:
kawalan ng balanse, kawalang-tatag, hindi pagkakatugma\nHalimbawa: Nahirapan siyang
hanapin ang balanse sa pagitan ng trabaho at personal na buhay.";
    e.synonyms = {"equilibrium", "stability", "symmetry"};
    e.antonyms = {"imbalance", "instability", "disproportion"};
    e.background = "Origin: Latin 'bilanx', having two pans";
    e.usage = "She struggled to find a balance between work and personal life.";
    m_words.append(e);

    e.word = "Banish";
    e.definition = "The king decided to banish the traitor from the kingdom.";
    e.translation = "Itaboy\nKasingkahulugan: ipatapon, paalisin, palayasin\nKasalungat:
tanggapin, anyayahan, papasukin\nHalimbawa: Nagpasya ang hari na itaboy ang taksil mula sa
kaharian.";
    e.synonyms = {"exile", "expel", "eject"};
    e.antonyms = {"welcome", "invite", "admit"};
    e.background = "Origin: Old French 'banir', to proclaim";
    e.usage = "The king decided to banish the traitor from the kingdom.";
    m_words.append(e);

    e.word = "Bare";
    e.definition = "The bare walls of the room made it feel cold and empty.";
    e.translation = "Hubad\nKasingkahulugan: walang takip, lantad, hubo\nKasalungat: may damit,
natatakpan, protektado\nHalimbawa: Ang hubad na mga pader ng silid ay nagbigay dito ng
malamig na pakiramdam.";
    e.synonyms = {"exposed", "uncovered", "nude"};
    e.antonyms = {"clothed", "covered", "protected"};
    e.background = "Origin: Old English 'baer', naked";
    e.usage = "The bare walls of the room made it feel cold and empty.";
    m_words.append(e);

    e.word = "Benevolent";
    e.definition = "The benevolent king always helped the poor.";
    e.translation = "Mapagkawanggawa\nKasingkahulugan: mabait, maawain,
matulungin\nKasalungat: malupit, masama, walang puso\nHalimbawa: Ang
mapagkawanggawang hari ay laging tumutulong sa mahihirap.";
    e.synonyms = {"kind", "charitable", "compassionate"};
    e.antonyms = {"malevolent", "unkind", "cruel"};
    e.background = "Origin: Latin 'benevolus', wishing well";
    e.usage = "The benevolent king always helped the poor.";
    m_words.append(e);

    e.word = "Brave";
    e.definition = "The brave firefighter ran into the burning building to rescue the family.";
    e.translation = "Matapang\nKasingkahulugan: walang takot, magiting, matibay ang
loob\nKasalungat: duwag, mahina ang loob, natatakot\nHalimbawa: Ang matapang na bumbero
ay pumasok sa nasusunog na gusali upang iligtas ang pamilya.";
    e.synonyms = {"courageous", "fearless", "valiant"};
    e.antonyms = {"cowardly", "timid", "fearful"};
    e.background = "Origin: Italian 'bravo', wild or fierce";
    e.usage = "The brave firefighter ran into the burning building to rescue the family.";
    m_words.append(e);

    e.word = "Bland";
    e.definition = "The soup was too bland for my taste; it needed more seasoning.";
    e.translation = "Maputla\nKasingkahulugan: matabang, walang lasa, walang sigla\nKasalungat:
malasa, maanghang, masigla\nHalimbawa: Masyadong maputla ang sabaw para sa panlasa ko;
kailangan pa ng pampalasa.";
    e.synonyms = {"dull", "tasteless", "insipid"};
    e.antonyms = {"flavorful", "spicy", "exciting"};
    e.background = "Origin: Latin 'blandus', smooth or flattering";
    e.usage = "The soup was too bland for my taste; it needed more seasoning.";
    m_words.append(e);

    e.word = "Blissful";
    e.definition = "They spent a blissful weekend in the mountains.";
    e.translation = "Masaya\nKasingkahulugan: maligaya, kuntento, masigla\nKasalungat:
malungkot, balisa, nagdadalamhati\nHalimbawa: Nagtamasa sila ng masayang weekend sa
kabundukan.";
    e.synonyms = {"happy", "content", "joyful"};
    e.antonyms = {"miserable", "unhappy", "sorrowful"};
    e.background = "Origin: Old English 'blisse', joy";
    e.usage = "They spent a blissful weekend in the mountains.";
    m_words.append(e);

    e.word = "Blunt";
    e.definition = "Her blunt response to the criticism surprised everyone in the room.";
    e.translation = "Diretso\nKasingkahulugan: prangka, tapat, walang paligoy\nKasalungat:
maingat, diplomatikong, maselan\nHalimbawa: Ang prangka niyang sagot sa puna ay ikinagulat
ng lahat sa silid.";
    e.synonyms = {"direct", "straightforward", "frank"};
    e.antonyms = {"tactful", "diplomatic", "subtle"};
    e.background = "Origin: Scandinavian 'blunt', dull or blunt edge";
    e.usage = "Her blunt response to the criticism surprised everyone in the room.";
    m_words.append(e);

    e.word = "Bore";
    e.definition = "The meeting lasted for hours and really started to bore me.";
    e.translation = "Nakakainip\nKasingkahulugan: nakakapagod, nakakaantok, walang
sigla\nKasalungat: nakakaaliw, nakakatawa, interesante\nHalimbawa: Napakatagal ng pulong at
talagang nakakainip.";
    e.synonyms = {"tire", "weary", "dull"};
    e.antonyms = {"entertain", "amuse", "interest"};
    e.background = "Origin: Old French 'borer', to drill";
    e.usage = "The meeting lasted for hours and really started to bore me.";
    m_words.append(e);

    e.word = "Brisk";
    e.definition = "She took a brisk walk in the park to get some fresh air.";
    e.translation = "Masigla\nKasingkahulugan: maliksi, mabilis, buhay na buhay\nKasalungat:
mabagal, matamlay, tamad\nHalimbawa: Naglakad siya nang masigla sa parke para makalanghap
ng sariwang hangin.";
    e.synonyms = {"energetic", "quick", "lively"};
    e.antonyms = {"sluggish", "slow", "lethargic"};
    e.background = "Origin: Scandinavian, meaning sharp or biting";
    e.usage = "She took a brisk walk in the park to get some fresh air.";
    m_words.append(e);
    e.word = "Bitter";
    e.definition = "The bitter argument left both of them upset and frustrated.";
    e.translation = "Mapait\nKasingkahulugan: marahas, masakit, may hinanakit\nKasalungat: 
matamis, magaan, maganda\nHalimbawa: Ang mapait na pagtatalo ay nagdulot ng sama ng loob 
sa magkabilang panig.";
    e.synonyms = {"harsh", "sour", "resentful"};
    e.antonyms = {"sweet", "pleasant", "mild"};
    e.background = "Origin: Old English 'biter', to bite";
    e.usage = "The bitter argument left both of them upset and frustrated.";
    m_words.append(e);

    e.word = "Bigotry";
    e.definition = "Bigotry has no place in a society that values equality.";
    e.translation = "Pagkapanatiko\nKasingkahulugan: pagkiling, pagtatangi, 
diskriminasyon\nKasalungat: pagiging bukas, pagtanggap, katarungan\nHalimbawa: Walang 
lugar ang pagkapanatiko sa lipunang nagmamahal sa pagkakapantay-pantay.";
    e.synonyms = {"intolerance", "prejudice", "discrimination"};
    e.antonyms = {"open-mindedness", "acceptance", "fairness"};
    e.background = "Origin: French 'bigoterie', derived from Bigos";
    e.usage = "Bigotry has no place in a society that values equality.";
    m_words.append(e);

    e.word = "Baffled";
    e.definition = "She was baffled by the strange behavior of her friend.";
    e.translation = "Nalito\nKasingkahulugan: naguluhan, nagtataka, nagtaka\nKasalungat: tiyak, 
malinaw, sigurado\nHalimbawa: Nalito siya sa kakaibang asal ng kanyang kaibigan.";
    e.synonyms = {"confused", "puzzled", "bewildered"};
    e.antonyms = {"certain", "clear", "sure"};
    e.background = "Origin: Scottish, meaning to check or repel";
    e.usage = "She was baffled by the strange behavior of her friend.";
    m_words.append(e);

    e.word = "Benevolence";
    e.definition = "His acts of benevolence made him beloved by all.";
    e.translation = "Kabaitan\nKasingkahulugan: kagandahang-loob, kabutihan, 
pagkamaawain\nKasalungat: kalupitan, kasamaan, pagiging makasarili\nHalimbawa: Ang 
kanyang kabaitan ay minahal ng lahat.";
    e.synonyms = {"kindness", "goodwill", "generosity"};
    e.antonyms = {"selfishness", "cruelty", "malevolence"};
    e.background = "Origin: Latin 'benevolentia', desire to do good";
    e.usage = "His acts of benevolence made him beloved by all.";
    m_words.append(e);

    e.word = "Brittle";
    e.definition = "The glass vase was brittle and broke into pieces with the slightest touch.";
    e.translation = "Marupok\nKasingkahulugan: madaling mabasag, mahina, 
maramdamin\nKasalungat: matibay, matatag, matigas\nHalimbawa: Ang basong iyon ay marupok
at agad nabasag nang mahulog.";
    e.synonyms = {"fragile", "delicate", "weak"};
    e.antonyms = {"durable", "strong", "resilient"};
    e.background = "Origin: Old English 'breotan', to break";
    e.usage = "The glass vase was brittle and broke into pieces with the slightest touch.";
    m_words.append(e);

    e.word = "Brilliant";
    e.definition = "The scientist's brilliant discovery changed the field forever.";
    e.translation = "Matalino\nKasingkahulugan: magaling, mahusay, henyo\nKasalungat: bobo,
mahina, walang alam\nHalimbawa: Ang matalinong tuklas ng siyentipiko ay nagbago ng larangan
magpakailanman.";
    e.synonyms = {"outstanding", "exceptional", "talented"};
    e.antonyms = {"dull", "mediocre", "uninspired"};
    e.background = "Origin: Italian 'brillare', to shine";
    e.usage = "The scientist's brilliant discovery changed the field forever.";
    m_words.append(e);

    e.word = "Bounty";
    e.definition = "The harvest provided a bounty of fruits and vegetables.";
    e.translation = "Kasaganaan\nKasingkahulugan: kayamanan, kasapatan,
kasiyahan\nKasalungat: kakulangan, kakapusan, paghihirap\nHalimbawa: Ang aning ito ay
nagdala ng kasaganaan ng prutas at gulay.";
    e.synonyms = {"abundance", "plenty", "wealth"};
    e.antonyms = {"scarcity", "shortage", "lack"};
    e.background = "Origin: Old French 'bonte', goodness";
    e.usage = "The harvest provided a bounty of fruits and vegetables.";
    m_words.append(e);

    e.word = "Blaze";
    e.definition = "The blaze of the campfire kept us warm on the cold night.";
    e.translation = "Apoy\nKasingkahulugan: siga, liyab, alab\nKasalungat: puksain, patayin,
apulahin\nHalimbawa: Ang apoy ng bonfire ang nagbigay-init sa malamig na gabi.";
    e.synonyms = {"fire", "flame", "inferno"};
    e.antonyms = {"extinguish", "douse", "put out"};
    e.background = "Origin: Old English 'blæse', white mark";
    e.usage = "The blaze of the campfire kept us warm on the cold night.";
    m_words.append(e);

    e.word = "Baffle";
    e.definition = "The magician's trick completely baffled the audience.";
    e.translation = "Malito\nKasingkahulugan: maguluhan, malabuan, mabigla\nKasalungat:
malinawan, maunawaan, masimplihan\nHalimbawa: Lubos na nalito ang mga tao sa mahikang
ipinakita ng salamangkero.";
    e.synonyms = {"confuse", "perplex", "bewilder"};
    e.antonyms = {"clarify", "explain", "simplify"};
    e.background = "Origin: Scottish, to check or repel";
    e.usage = "The magician's trick completely baffled the audience.";
    m_words.append(e);

    e.word = "Brawl";
    e.definition = "The two men got into a brawl outside the bar.";
    e.translation = "Gulo\nKasingkahulugan: suntukan, sagupaan, alitan\nKasalungat:
kapayapaan, pagkakaisa, katahimikan\nHalimbawa: Nagkaroon ng gulo sa labas ng bar dahil sa
dalawang lalaki.";
    e.synonyms = {"fight", "clash", "scuffle"};
    e.antonyms = {"peace", "harmony", "calm"};
    e.background = "Origin: Middle Dutch 'bralle', to brawl";
    e.usage = "The two men got into a brawl outside the bar.";
    m_words.append(e);

    e.word = "Bright";
    e.definition = "The future looks bright for young professionals in this field.";
    e.translation = "Maliwanag\nKasingkahulugan: maningning, makintab,
kumikislap\nKasalungat: madilim, mapusyaw, walang ilaw\nHalimbawa: Maliwanag ang
kinabukasan ng mga kabataang propesyonal sa larangang ito.";
    e.synonyms = {"radiant", "brilliant", "shining"};
    e.antonyms = {"dull", "dim", "dark"};
    e.background = "Origin: Old English 'beorht', shining";
    e.usage = "The future looks bright for young professionals in this field.";
    m_words.append(e);

    e.word = "Blunder";
    e.definition = "He made a huge blunder during the presentation by forgetting his key points.";
    e.translation = "Pagkakamali\nKasingkahulugan: error, pagkukulang, pagkadulas\nKasalungat:
tagumpay, tama, katuparan\nHalimbawa: Nagkaroon siya ng malaking pagkakamali sa
presentasyon nang makalimutan ang mahahalagang punto.";
    e.synonyms = {"mistake", "error", "misstep"};
    e.antonyms = {"success", "achievement", "triumph"};
    e.background = "Origin: Scandinavian 'blunda', to doze";
    e.usage = "He made a huge blunder during the presentation by forgetting his key points.";
    m_words.append(e);

    e.word = "Bizarre";
    e.definition = "His bizarre behavior left everyone in the office confused.";
    e.translation = "Kakaiba\nKasingkahulugan: pambihira, hindi karaniwan,
nakapagtataka\nKasalungat: normal, karaniwan, pangkaraniwan\nHalimbawa: Ang kakaibang
kilos niya ay nakagulat sa lahat sa opisina.";
    e.synonyms = {"strange", "odd", "peculiar"};
    e.antonyms = {"normal", "conventional", "typical"};
    e.background = "Origin: Spanish 'bizarro', brave or fierce";
```

```
   e.usage = "His bizarre behavior left everyone in the office confused.";
   m_words.append(e);

   e.word = "Breezy";
   e.definition = "The breezy afternoon made the beach a perfect spot to relax.";
   e.translation = "Mahangin\nKasingkahulugan: presko, maaliwalas, may simoy\nKasalungat:
kalmado, walang hangin, tahimik\nHalimbawa: Ang mahangin na hapon ay perpekto para
magpahinga sa tabing-dagat.";
   e.synonyms = {"windy", "fresh", "airy"};
   e.antonyms = {"calm", "still", "quiet"};
   e.background = "Origin: English 'breeze', light wind";
   e.usage = "The breezy afternoon made the beach a perfect spot to relax.";
   m_words.append(e);

   e.word = "Bumpy";
   e.definition = "The road was bumpy, making the ride uncomfortable.";
   e.translation = "Lubak-lubak\nKasingkahulugan: hindi pantay, magaspang,
mabundok\nKasalungat: makinis, pantay, maayos\nHalimbawa: Lubak-lubak ang kalsada kaya
hindi komportable ang biyahe.";
   e.synonyms = {"uneven", "rough", "rugged"};
   e.antonyms = {"smooth", "level", "even"};
   e.background = "Origin: English 'bump', a raised mass";
   e.usage = "The road was bumpy, making the ride uncomfortable.";
   m_words.append(e);

   e.word = "Boost";
   e.definition = "The new advertising campaign helped boost sales.";
   e.translation = "Palakasin\nKasingkahulugan: dagdagan, paunlarin, pataasin\nKasalungat:
bawasan, pahinain, pababain\nHalimbawa: Ang bagong kampanya sa marketing ay nakatulong
para palakasin ang benta.";
   e.synonyms = {"increase", "raise", "enhance"};
   e.antonyms = {"decrease", "diminish", "reduce"};
   e.background = "Origin: English 'boost', to push up";
   e.usage = "The new advertising campaign helped boost sales.";
   m_words.append(e);

   e.word = "Bold";
   e.definition = "His bold decision to start a new business paid off in the end.";
   e.translation = "Matapang\nKasingkahulugan: walang takot, palaban,
mapangahas\nKasalungat: mahina, duwag, matatakutin\nHalimbawa: Ang matapang niyang
desisyon na magsimula ng negosyo ay nagbunga ng tagumpay.";
   e.synonyms = {"daring", "courageous", "fearless"};
   e.antonyms = {"timid", "cautious", "afraid"};
   e.background = "Origin: Old English 'bald', confident";
   e.usage = "His bold decision to start a new business paid off in the end.";
   m_words.append(e);
```

```
    e.word = "Bashful";
    e.definition = "The bashful child hid behind his mother when meeting strangers.";
    e.translation = "Mahiyain\nKasingkahulugan: mailap, tahimik, mahina ang loob\nKasalungat:
palakaibigan, kumpiyansa, matapang\nHalimbawa: Ang mahiyain na bata ay nagtago sa likod ng
kanyang ina nang makilala ang bagong tao.";
    e.synonyms = {"shy", "timid", "self-conscious"};
    e.antonyms = {"outgoing", "confident", "bold"};
    e.background = "Origin: English 'bash', to strike";
    e.usage = "The bashful child hid behind his mother when meeting strangers.";
    m_words.append(e);

    e.word = "Beaming";
    e.definition = "She walked into the room with a beaming smile on her face.";
    e.translation = "Nakangiti\nKasingkahulugan: masigla, maliwanag, masaya\nKasalungat:
malungkot, matamlay, seryoso\nHalimbawa: Pumasok siya sa silid na may nakangiting mukha.";
    e.synonyms = {"radiant", "glowing", "cheerful"};
    e.antonyms = {"gloomy", "sad", "downcast"};
    e.background = "Origin: English 'beam', a ray of light";
    e.usage = "She walked into the room with a beaming smile on her face.";
    m_words.append(e);

    e.word = "Bountiful";
    e.definition = "The garden produced a bountiful harvest this year.";
    e.translation = "Masagana\nKasingkahulugan: marami, sagana, mapagbigay\nKasalungat:
kulang, kakaunti, limitado\nHalimbawa: Ang hardin ay nagbunga ng masaganang ani ngayong
taon.";
    e.synonyms = {"plentiful", "abundant", "generous"};
    e.antonyms = {"scarce", "insufficient", "limited"};
    e.background = "Origin: Old French 'bonte', goodness";
    e.usage = "The garden produced a bountiful harvest this year.";
    m_words.append(e);

    e.word = "Brutal";
    e.definition = "The brutal truth was hard to hear but necessary.";
    e.translation = "Malupit\nKasingkahulugan: marahas, walang awa, brutal\nKasalungat: mabait,
maawain, banayad\nHalimbawa: Mahirap tanggapin ang malupit na katotohanan, ngunit ito ay
kailangan.";
    e.synonyms = {"savage", "cruel", "harsh"};
    e.antonyms = {"gentle", "kind", "compassionate"};
    e.background = "Origin: Latin 'brutus', dull or stupid";
    e.usage = "The brutal truth was hard to hear but necessary.";
    m_words.append(e);

    e.word = "Befriend";
    e.definition = "He tried to befriend the new student by offering help with her homework.";
    e.translation = "Makipagkaibigan\nKasingkahulugan: tumulong, sumuporta, makisama\nKasalungat: lumaban, tumanggi, kontrahin\nHalimbawa: Sinubukan niyang makipagkaibigan sa bagong estudyante sa pamamagitan ng pagtulong sa takdang-aralin.";
    e.synonyms = {"ally", "support", "assist"};
    e.antonyms = {"antagonize", "reject", "oppose"};
    e.background = "Origin: English 'friend', a person one knows well";
    e.usage = "He tried to befriend the new student by offering help with her homework.";
    m_words.append(e);

    e.word = "Bliss";
    e.definition = "They lived in bliss for many years after their wedding.";
    e.translation = "Kaligayahan\nKasingkahulugan: saya, tuwa, kasiyahan\nKasalungat: kalungkutan, dalamhati, pighati\nHalimbawa: Nabuhay sila sa kaligayahan matapos ang kanilang kasal.";
    e.synonyms = {"happiness", "joy", "contentment"};
    e.antonyms = {"misery", "sorrow", "sadness"};
    e.background = "Origin: Old English 'blisse', joy";
    e.usage = "They lived in bliss for many years after their wedding.";
    m_words.append(e);

    e.word = "Bash";
    e.definition = "He gave the door a bash with the hammer, trying to fix it.";
    e.translation = "Hampas\nKasingkahulugan: suntok, palo, bugbog\nKasalungat: tapik, tulak, dampi\nHalimbawa: Hinampas niya ang pinto gamit ang martilyo upang ayusin ito.";
    e.synonyms = {"strike", "hit", "slam"};
    e.antonyms = {"tap", "poke", "nudge"};
    e.background = "Origin: Scandinavian 'base', a blow";
    e.usage = "He gave the door a bash with the hammer, trying to fix it.";
    m_words.append(e);

    e.word = "Ban";
    e.definition = "The school decided to ban cell phones during class.";
    e.translation = "Ipinagbawal\nKasingkahulugan: ipagbawal, ipatigil, ipahinto\nKasalungat: payagan, pahintulutan, pahintulot\nHalimbawa: Nagpasya ang paaralan na ipagbawal ang paggamit ng cellphone sa klase.";
    e.synonyms = {"prohibit", "forbid", "outlaw"};
    e.antonyms = {"allow", "permit", "authorize"};
    e.background = "Origin: Old Norse 'banna', to forbid";
    e.usage = "The school decided to ban cell phones during class.";
    m_words.append(e);

    e.word = "Befuddle";
    e.definition = "The complicated instructions befuddled the new employees.";
    e.translation = "Lituhin\nKasingkahulugan: guluhin, lituhin, gulantangin\nKasalungat: ipaliwanag, linawin, payapain\nHalimbawa: Nalito ang mga bagong empleyado sa komplikadong tagubilin.";
    e.synonyms = {"confuse", "perplex", "bewilder"};
    e.antonyms = {"clarify", "explain", "simplify"};
    e.background = "Origin: English 'befuddle', to confuse utterly";
    e.usage = "The complicated instructions befuddled the new employees.";
    m_words.append(e);

    e.word = "Bristle";
    e.definition = "His anger made his hair bristle with frustration.";
    e.translation = "Tindig-Balahibo\nKasingkahulugan: tumayo, manigas, magalit\nKasalungat:
mag-relax, lumambot, kukalma\nHalimbawa: Tumindig ang kanyang balahibo sa galit.";
    e.synonyms = {"stiffen", "stand up", "flare"};
    e.antonyms = {"relax", "soften", "calm"};
    e.background = "Origin: Old English 'byrst', to burst";
    e.usage = "His anger made his hair bristle with frustration.";
    m_words.append(e);

    e.word = "Banishment";
    e.definition = "The punishment for breaking the rules was banishment.";
    e.translation = "Pagkatapon\nKasingkahulugan: pagpapatalsik, pagpapaalis,
deportasyon\nKasalungat: pagbabalik, pagtanggap, pagbabalik-tanaw\nHalimbawa: Ang parusa
sa krimen ay pagkatapon sa komunidad.";
    e.synonyms = {"exile", "expulsion", "deportation"};
    e.antonyms = {"admission", "welcome", "acceptance"};
    e.background = "Origin: Old French 'banir', to proclaim";
    e.usage = "The punishment for breaking the rules was banishment.";
    m_words.append(e);

    e.word = "Bait";
    e.definition = "The fisherman used worms as bait to catch the fish.";
    e.translation = "Pang-akit\nKasingkahulugan: pain, tukso, panghila\nKasalungat: panglayo,
pangtaboy, pang-iwas\nHalimbawa: Gumamit ang mangingisda ng bulate bilang pain sa
pangingisda.";
    e.synonyms = {"lure", "entice", "attract"};
    e.antonyms = {"discourage", "repel", "deter"};
    e.background = "Origin: Old Norse 'beita', to feed";
    e.usage = "The fisherman used worms as bait to catch the fish.";
    m_words.append(e);

    e.word = "Braggart";
    e.definition = "He's such a braggart that no one likes to talk to him.";
    e.translation = "Mayabang\nKasingkahulugan: palalo, mapagmataas,
nagyayabang\nKasalungat: mapagkumbaba, mahinhin, maamo\nHalimbawa: Napakayabang niya
kaya walang gustong makipag-usap sa kanya.";
    e.synonyms = {"boaster", "show-off", "egotist"};
    e.antonyms = {"humble", "modest", "reserved"};
    e.background = "Origin: Old French 'braguete', boasting";
    e.usage = "He's such a braggart that no one likes to talk to him.";
    m_words.append(e);

    e.word = "Befit";
    e.definition = "The luxury hotel was a perfect place to befit her status.";
    e.translation = "Bagay\nKasingkahulugan: akma, nararapat, tugma\nKasalungat: hindi tugma,
salungat, taliwas\nHalimbawa: Ang marangyang hotel ay bagay sa kanyang katayuan.";
    e.synonyms = {"suit", "be appropriate", "fit"};
    e.antonyms = {"misfit", "clash", "mismatch"};
    e.background = "Origin: Old English 'befittan', to make suitable";
    e.usage = "The luxury hotel was a perfect place to befit her status.";
    m_words.append(e);

    e.word = "Breach";
    e.definition = "The company was sued for a breach of contract.";
    e.translation = "Paglabag\nKasingkahulugan: pagkakasala, pagsuway, paglabag\nKasalungat:
pagsunod, pagtalima, paggalang\nHalimbawa: Dinimanda ang kompanya dahil sa paglabag sa
kontrata.";
    e.synonyms = {"violation", "infraction", "break"};
    e.antonyms = {"compliance", "observance", "respect"};
    e.background = "Origin: Old French 'breche', a break";
    e.usage = "The company was sued for a breach of contract.";
    m_words.append(e);

    e.word = "Bellow";
    e.definition = "He began to bellow in frustration when he couldn't find the keys.";
    e.translation = "Sigaw\nKasingkahulugan: hiyaw, bulalas, alulong\nKasalungat: bulong, ungol,
pabulong\nHalimbawa: Napasigaw siya sa galit nang hindi makita ang kanyang susi.";
    e.synonyms = {"shout", "yell", "roar"};
    e.antonyms = {"whisper", "murmur", "mutter"};
    e.background = "Origin: Old English 'belgan', to swell";
    e.usage = "He began to bellow in frustration when he couldn't find the keys.";
    m_words.append(e);

    e.word = "Betray";
    e.definition = "He felt heartbroken after his best friend betrayed him.";
    e.translation = "Ipagkanulo\nKasingkahulugan: linlangin, traydurin, lokohin\nKasalungat:
tulungan, ipagtanggol, maging tapat\nHalimbawa: Labis ang kanyang sama ng loob nang siya'y
ipagkanulo ng matalik na kaibigan.";
    e.synonyms = {"deceive", "backstab", "mislead"};
    e.antonyms = {"support", "stand by", "be loyal"};
    e.background = "Origin: Old French 'betrayer', to deliver up";
    e.usage = "He felt heartbroken after his best friend betrayed him.";
    m_words.append(e);

    e.word = "Baggage";
    e.definition = "She packed all her baggage before heading to the airport.";
    e.translation = "Bagahe\nKasingkahulugan: maleta, gamit, dalahin\nKasalungat: wala (depende
sa gamit)\nHalimbawa: Ipinakete niya ang lahat ng kanyang bagahe bago pumunta sa paliparan.";
    e.synonyms = {"luggage", "suitcases", "possessions"};
    e.antonyms = {};
    e.background = "Origin: Old French 'bagage', what is carried";
    e.usage = "She packed all her baggage before heading to the airport.";
    m_words.append(e);

    e.word = "Bully";
    e.definition = "He became the target of a bully at school who took his lunch money.";
    e.translation = "Mang-api\nKasingkahulugan: manakot, mang-asar, manggulpi\nKasalungat:
ipagtanggol, protektahan, tulungan\nHalimbawa: Naging biktima siya ng mang-aapi na kumukuha
ng kanyang baon sa paaralan.";
    e.synonyms = {"intimidate", "harass", "persecute"};
    e.antonyms = {"protect", "defend", "support"};
    e.background = "Origin: Dutch 'boel', lover or brother";
    e.usage = "He became the target of a bully at school who took his lunch money.";
    m_words.append(e);

    // Letter C Words
    // ---------------------------------------------------------------
    e.word = "Courageous";
    e.definition = "The courageous soldier saved his comrades under heavy fire.";
    e.translation = "Matapang\nKasingkahulugan: matapang, magiting, walang takot\nKasalungat:
duwag, takot, mahiyain\nHalimbawa: Ang matapang na sundalo ay iniligtas ang kanyang mga
kasama sa gitna ng putukan.";
    e.synonyms = {"brave", "valiant", "fearless"};
    e.antonyms = {"cowardly", "fearful", "timid"};
    e.background = "Origin: Latin 'coraticus' (via Old French), relating to bravery";
    e.usage = "The courageous soldier saved his comrades under heavy fire.";
    m_words.append(e);

    e.word = "Clever";
    e.definition = "His clever solution to the problem impressed everyone.";
    e.translation = "Matalino\nKasingkahulugan: matalino, madiskarte, marunong\nKasalungat:
mangmang, hangal, inosente\nHalimbawa: Ang matalinong solusyon niya sa problema ay
ikinamangha ng lahat.";
    e.synonyms = {"smart", "witty", "intelligent"};
    e.antonyms = {"dumb", "foolish", "naive"};
    e.background = "Origin: Old English 'clǣfre', quick to understand";
    e.usage = "His clever solution to the problem impressed everyone.";
    m_words.append(e);

    e.word = "Clumsy";
    e.definition = "She felt clumsy as she tripped over the chair.";
    e.translation = "Tulad ng walang kilos\nKasingkahulugan: awkward, uncoordinated,
graceless\nKasalungat: graceful, coordinated, agile\nHalimbawa: Nahulog siya sa upuan dahil sa
pagiging clumsy niya.";
    e.synonyms = {"awkward", "uncoordinated", "graceless"};
    e.antonyms = {"graceful", "coordinated", "agile"};
    e.background = "Origin: uncertain, related to lacking coordination";
    e.usage = "She felt clumsy as she tripped over the chair.";
    m_words.append(e);

    e.word = "Cautious";
    e.definition = "The cautious driver slowed down when the weather became foggy.";
    e.translation = "Maingat\nKasingkahulugan: maingat, mapanuri, mapagmatyag\nKasalungat:
pabaya, padalos-dalos, walang ingat\nHalimbawa: Ang maingat na drayber ay bumagal nang
maging mahamog ang daan.";
    e.synonyms = {"careful", "prudent", "wary"};
    e.antonyms = {"reckless", "careless", "hasty"};
    e.background = "Origin: Latin 'cautus', careful";
    e.usage = "The cautious driver slowed down when the weather became foggy.";
    m_words.append(e);

    e.word = "Charming";
    e.definition = "He was a charming host who made everyone feel welcome.";
    e.translation = "Kaakit-akit\nKasingkahulugan: nakakaaliw, kahali-halina, may
karisma\nKasalungat: hindi kaaya-aya, bastos, walang dating\nHalimbawa: Siya ay isang
kaakit-akit na host na nagparamdam ng ginhawa sa lahat.";
    e.synonyms = {"delightful", "enchanting", "charismatic"};
    e.antonyms = {"unappealing", "unattractive", "rude"};
    e.background = "Origin: Old French 'charmant', to enchant";
    e.usage = "He was a charming host who made everyone feel welcome.";
    m_words.append(e);

    e.word = "Curious";
    e.definition = "The child was curious about the world around him and asked a lot of
questions.";
    e.translation = "Palaisip\nKasingkahulugan: mausisa, interesado, sabik matuto\nKasalungat:
walang pakialam, hindi interesado, malamig\nHalimbawa: Ang bata ay palaisip tungkol sa mundo
at madalas magtanong.";
    e.synonyms = {"inquisitive", "interested", "eager to learn"};
    e.antonyms = {"indifferent", "uninterested", "apathetic"};
    e.background = "Origin: Latin 'curiosus', inquisitive";
    e.usage = "The child was curious about the world around him and asked a lot of questions.";
    m_words.append(e);

    e.word = "Chilly";
    e.definition = "It was a chilly morning, so I grabbed my jacket before heading out.";
    e.translation = "Malamig\nKasingkahulugan: malamig, presko, maginaw\nKasalungat: mainit,
maalinsangan, maalab\nHalimbawa: Maginaw ang umaga kaya nagsuot ako ng jacket bago
umalis.";
    e.synonyms = {"cold", "cool", "brisk"};
    e.antonyms = {"warm", "hot", "toasty"};
    e.background = "Origin: Old English 'ciele', chilly";
    e.usage = "It was a chilly morning, so I grabbed my jacket before heading out.";
    m_words.append(e);

    e.word = "Courage";
    e.definition = "It took a lot of courage to speak in front of such a large crowd.";
    e.translation = "Tapang\nKasingkahulugan: bravery, fortitude, valor\nKasalungat: fear,
cowardice, timidity\nHalimbawa: Kinailangan ng malaking tapang para magsalita sa harap ng
malaking madla.";
    e.synonyms = {"bravery", "fortitude", "valor"};
    e.antonyms = {"fear", "cowardice", "timidity"};
    e.background = "Origin: Latin 'coraticus' via Old French";
    e.usage = "It took a lot of courage to speak in front of such a large crowd.";
    m_words.append(e);

    e.word = "Cynical";
    e.definition = "His cynical attitude made it hard for him to believe in others' goodwill.";
    e.translation = "Mapangduda\nKasingkahulugan: distrustful, skeptical, doubtful\nKasalungat:
trusting, hopeful, optimistic\nHalimbawa: Dahil sa kanyang mapangdudang ugali, hirap siyang
maniwala sa kabutihan ng iba.";
    e.synonyms = {"distrustful", "skeptical", "doubtful"};
    e.antonyms = {"trusting", "hopeful", "optimistic"};
    e.background = "Origin: Greek 'kynikos', dog-like";
    e.usage = "His cynical attitude made it hard for him to believe in others' goodwill.";
    m_words.append(e);

    e.word = "Compassionate";
    e.definition = "She was a compassionate nurse who always took extra time with her patients.";
    e.translation = "Maawain\nKasingkahulugan: empathetic, kind, caring\nKasalungat: indifferent,
apathetic, callous\nHalimbawa: Siya ay isang maawain na nars na laging nagbibigay ng oras sa
mga pasyente.";
    e.synonyms = {"empathetic", "kind", "caring"};
    e.antonyms = {"indifferent", "apathetic", "callous"};
    e.background = "Origin: Latin 'compassio', to suffer with";
    e.usage = "She was a compassionate nurse who always took extra time with her patients.";
    m_words.append(e);

    e.word = "Confident";
    e.definition = "She walked into the meeting with a confident attitude.";
    e.translation = "May Kumpiyansa\nKasingkahulugan: self-assured, assertive,
positive\nKasalungat: insecure, uncertain, unsure\nHalimbawa: Lumakad siya sa pulong nang
may kumpiyansang ugali.";
```

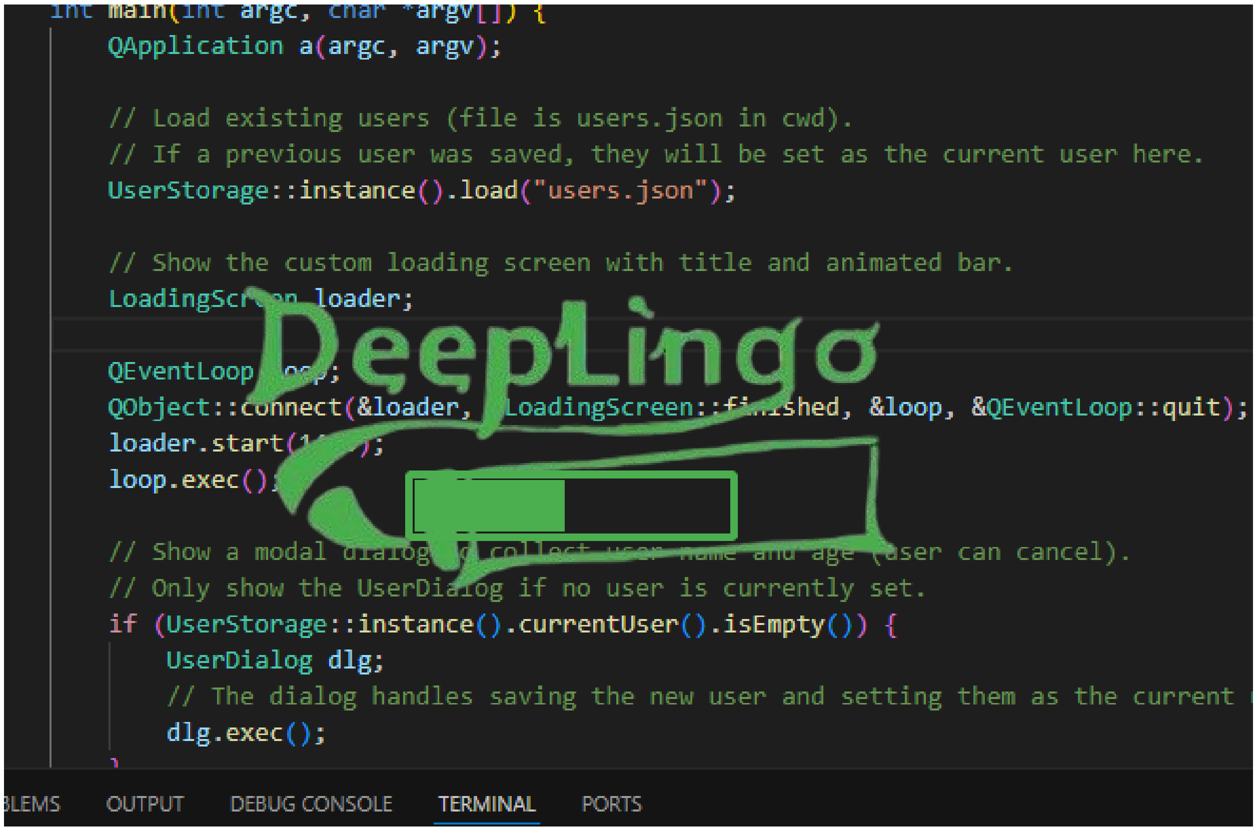
```
    e.synonyms = {"self-assured", "assertive", "positive"};
    e.antonyms = {"insecure", "uncertain", "unsure"};
    e.background = "Origin: Latin 'confidere', to trust";
    e.usage = "She walked into the meeting with a confident attitude.";
    m_words.append(e);

    e.word = "Complacent";
    e.definition = "He became complacent after achieving success and stopped working hard.";
    e.translation = "Kuntento sa Sarili\nKasingkahulugan: self-satisfied, content,
untroubled\nKasalungat: ambitious, dissatisfied, restless\nHalimbawa: Naging kampante siya
matapos magtagumpay at tumigil sa pagsisikap.";
    e.synonyms = {"self-satisfied", "content", "untroubled"};
    e.antonyms = {"ambitious", "dissatisfied", "restless"};
    e.background = "Origin: Latin 'complacere', to please";
    e.usage = "He became complacent after achieving success and stopped working hard.";
    m_words.append(e);

    e.word = "Chaotic";
    e.definition = "The streets were chaotic after the parade, with people everywhere.";
    e.translation = "Magulo\nKasingkahulugan: disorderly, tumultuous, confusing\nKasalungat:
orderly, organized, calm\nHalimbawa: Magulo ang mga kalsada matapos ang parada, puno ng tao
sa lahat ng dako.";
    e.synonyms = {"disorderly", "tumultuous", "confusing"};
    e.antonyms = {"orderly", "organized", "calm"};
    e.background = "Origin: Greek 'chaos'";
    e.usage = "The streets were chaotic after the parade, with people everywhere.";
    m_words.append(e);

    e.word = "Cumbersome";
    e.definition = "The cumbersome package was hard to carry up the stairs.";
    e.translation = "Mabigat Dalhin\nKasingkahulugan: awkward, unwieldy, bulky\nKasalungat:
manageable, easy, simple\nHalimbawa: Ang mabigat na pakete ay mahirap buhatin paakyat ng
hagdan.";
    e.synonyms = {"awkward", "unwieldy", "bulky"};
    e.antonyms = {"manageable", "easy", "simple"};
    e.background = "Origin: Old Norse/Old English roots relating to burden";
    e.usage = "The cumbersome package was hard to carry up the stairs.";
    m_words.append(e);

    e.word = "Cautiously";
    e.definition = "He moved cautiously around the broken glass on the floor.";
    e.translation = "Maingat na Paraan\nKasingkahulugan: carefully, warily, prudently\nKasalungat:
recklessly, hastily, carelessly\nHalimbawa: Maingat siyang gumalaw sa paligid ng nabasag na
baso sa sahig.";
    e.synonyms = {"carefully", "warily", "prudently"};
    e.antonyms = {"recklessly", "hastily", "carelessly"};
    e.background = "Adverbial form of cautious";
    e.usage = "He moved cautiously around the broken glass on the floor.";
    m_words.append(e);

    e.word = "Crucial";
    e.definition = "It is crucial to follow the safety instructions when operating heavy machinery.";
    e.translation = "Mahalaga\nKasingkahulugan: critical, vital, essential\nKasalungat: trivial,
insignificant, unimportant\nHalimbawa: Mahalagang sundin ang mga panuntunan sa kaligtasan
kapag gumagamit ng mabigat na makina.";
    e.synonyms = {"critical", "vital", "essential"};
    e.antonyms = {"trivial", "insignificant", "unimportant"};
    e.background = "Origin: Greek 'krisis', decisive moment";
    e.usage = "It is crucial to follow the safety instructions when operating heavy machinery.";
    m_words.append(e);

    e.word = "Cleverness";
    e.definition = "Her cleverness in solving the riddle impressed everyone at the party.";
    e.translation = "Katalinuhan\nKasingkahulugan: ingenuity, intelligence, wit\nKasalungat:
stupidity, dullness, clumsiness\nHalimbawa: Ang kanyang katalinuhan sa paglutas ng palaisipan
ay ikinamangha ng lahat.";
    e.synonyms = {"ingenuity", "intelligence", "wit"};
    e.antonyms = {"stupidity", "dullness", "clumsiness"};
    e.background = "Abstract noun from clever";
    e.usage = "Her cleverness in solving the riddle impressed everyone at the party.";
    m_words.append(e);

    e.word = "Conservative";
    e.definition = "The conservative approach to the project emphasized safety and stability.";
    e.translation = "Konserbatibo\nKasingkahulugan: traditional, cautious,
conventional\nKasalungat: liberal, progressive, radical\nHalimbawa: Ang konserbatibong paraan
ng proyekto ay nakatuon sa kaligtasan at katatagan.";
    e.synonyms = {"traditional", "cautious", "conventional"};
    e.antonyms = {"liberal", "progressive", "radical"};
    e.background = "Origin: Latin 'conservare', to preserve";
    e.usage = "The conservative approach to the project emphasized safety and stability.";
    m_words.append(e);

    e.word = "Contradictory";
    e.definition = "His contradictory statements left everyone confused about his true intentions.";
    e.translation = "Magkasalungat\nKasingkahulugan: inconsistent, conflicting,
opposing\nKasalungat: consistent, harmonious, matching\nHalimbawa: Ang kanyang
magkasalungat na pahayag ay nagdulot ng kalituhan.";
    e.synonyms = {"inconsistent", "conflicting", "opposing"};
    e.antonyms = {"consistent", "harmonious", "matching"};
    e.background = "From contra- + dictate, opposing";
    e.usage = "His contradictory statements left everyone confused about his true intentions.";
    m_words.append(e);

    e.word = "Crisis";
    e.definition = "The company faced a crisis after a major financial loss.";
    e.translation = "Krisis\nKasingkahulugan: emergency, disaster, catastrophe\nKasalungat:
solution, recovery, resolution\nHalimbawa: Nakaranas ng krisis ang kumpanya matapos ang
malaking pagkalugi.";
    e.synonyms = {"emergency", "disaster", "catastrophe"};
    e.antonyms = {"solution", "recovery", "resolution"};
    e.background = "Origin: Greek 'krisis', decision";
    e.usage = "The company faced a crisis after a major financial loss.";
    m_words.append(e);

    e.word = "Competent";
    e.definition = "She is a highly competent manager who always gets the job done.";
    e.translation = "Magaling\nKasingkahulugan: capable, skilled, proficient\nKasalungat:
incompetent, unskilled, inept\nHalimbawa: Isa siyang mahusay na tagapamahala na laging
natatapos ang gawain.";
    e.synonyms = {"capable", "skilled", "proficient"};
    e.antonyms = {"incompetent", "unskilled", "inept"};
    e.background = "From Latin 'competentia'";
    e.usage = "She is a highly competent manager who always gets the job done.";
    m_words.append(e);

    e.word = "Crude";
    e.definition = "His crude humor made some people uncomfortable at the dinner table.";
    e.translation = "Bastos\nKasingkahulugan: unrefined, rough, vulgar\nKasalungat: refined,
sophisticated, polite\nHalimbawa: Ang kanyang bastos na biro ay naka-offend sa ilan sa
hapag-kainan.";
    e.synonyms = {"unrefined", "rough", "vulgar"};
    e.antonyms = {"refined", "sophisticated", "polite"};
    e.background = "Origin: Old English 'cruden', raw";
    e.usage = "His crude humor made some people uncomfortable at the dinner table.";
    m_words.append(e);

    e.word = "Calm";
    e.definition = "The calm waters of the lake reflected the evening sky beautifully.";
    e.translation = "Kalma\nKasingkahulugan: peaceful, serene, composed\nKasalungat: agitated,
nervous, anxious\nHalimbawa: Ang kalmadong tubig ng lawa ay maganda sa ilalim ng langit sa
gabi.";
    e.synonyms = {"peaceful", "serene", "composed"};
    e.antonyms = {"agitated", "nervous", "anxious"};
    e.background = "Origin: Old English 'calm', tranquil";
    e.usage = "The calm waters of the lake reflected the evening sky beautifully.";
    m_words.append(e);
    e.word = "Cleverly";
    e.definition = "He cleverly avoided the question by changing the topic.";
    e.translation = "Matalinong Paraan\nKasingkahulugan: intelligently, wittily,
astutely\nKasalungat: foolishly, ineptly, clumsily\nHalimbawa: Matalino niyang naiwasan ang
tanong sa pamamagitan ng pagbabago ng paksa.";
    e.synonyms = {"intelligently", "wittily", "astutely"};
    e.antonyms = {"foolishly", "ineptly", "clumsily"};
    e.background = "Adverbial form of clever";
    e.usage = "He cleverly avoided the question by changing the topic.";
    m_words.append(e);

    e.word = "Culminate";
    e.definition = "The event will culminate with a grand fireworks display.";
    e.translation = "Umuabot sa Sukdulan\nKasingkahulugan: conclude, finish,
climax\nKasalungat: begin, initiate, start\nHalimbawa: Ang kaganapan ay magtatapos sa isang
engrandeng paputok.";
    e.synonyms = {"conclude", "finish", "climax"};
    e.antonyms = {"begin", "initiate", "start"};
    e.background = "From Latin 'culminare', to summit";
    e.usage = "The event will culminate with a grand fireworks display.";
    m_words.append(e);

    e.word = "Challenging";
    e.definition = "The math problem was challenging, but she solved it after a few tries.";
    e.translation = "Mahamon\nKasingkahulugan: difficult, demanding, tough\nKasalungat: easy,
simple, effortless\nHalimbawa: Mahirap ang problema sa math, pero nalutas niya pagkatapos ng
ilang ulit.";
    e.synonyms = {"difficult", "demanding", "tough"};
    e.antonyms = {"easy", "simple", "effortless"};
    e.background = "Modern English usage";
    e.usage = "The math problem was challenging, but she solved it after a few tries.";
    m_words.append(e);

    e.word = "Corrupt";
    e.definition = "The corrupt officials were arrested after an investigation uncovered their
crimes.";
    e.translation = "Tiwali\nKasingkahulugan: dishonest, immoral, depraved\nKasalungat: honest,
virtuous, moral\nHalimbawa: Ang mga tiwaling opisyal ay nahuli matapos ang masusing
imbestigasyon.";
    e.synonyms = {"dishonest", "immoral", "depraved"};
    e.antonyms = {"honest", "virtuous", "moral"};
    e.background = "From Latin 'corrumpere', to destroy";
    e.usage = "The corrupt officials were arrested after an investigation uncovered their crimes.";
    m_words.append(e);

    e.word = "Conducive";
    e.definition = "The quiet room was conducive to studying and concentration.";
    e.translation = "Nakakatulong\nKasingkahulugan: favorable, helpful,
advantageous\nKasalungat: harmful, obstructive, detrimental\nHalimbawa: Ang tahimik na silid
ay nakakatulong sa pag-aaral at konsentrasyon.";
    e.synonyms = {"favorable", "helpful", "advantageous"};
    e.antonyms = {"harmful", "obstructive", "detrimental"};
    e.background = "From Latin 'conducere', to lead together";
    e.usage = "The quiet room was conducive to studying and concentration.";
    m_words.append(e);

    e.word = "Contentious";
    e.definition = "The meeting became contentious as both sides refused to compromise.";
    e.translation = "Mapagtalo\nKasingkahulugan: argumentative, controversial,
combative\nKasalungat: agreeable, peaceful, harmonious\nHalimbawa: Naging mainit ang
pagtatalo dahil parehong ayaw magpatalo ang magkabilang panig.";
    e.synonyms = {"argumentative", "controversial", "combative"};
    e.antonyms = {"agreeable", "peaceful", "harmonious"};
    e.background = "From Latin 'contentio', dispute";
    e.usage = "The meeting became contentious as both sides refused to compromise.";
    m_words.append(e);

    e.word = "Complicated";
    e.definition = "The complicated instructions confused everyone trying to assemble the
furniture.";
    e.translation = "Kumplikado\nKasingkahulugan: complex, intricate, convoluted\nKasalungat:
simple, straightforward, clear\nHalimbawa: Nalito ang lahat sa komplikadong tagubilin sa pagbuo
ng muwebles.";
    e.synonyms = {"complex", "intricate", "convoluted"};
    e.antonyms = {"simple", "straightforward", "clear"};
    e.background = "From Latin 'complicare', to fold together";
    e.usage = "The complicated instructions confused everyone trying to assemble the furniture.";
    m_words.append(e);

    e.word = "Critical";
    e.definition = "Your critical feedback helped improve the quality of the final report.";
    e.translation = "Kritikal\nKasingkahulugan: essential, crucial, urgent\nKasalungat:
insignificant, trivial, unimportant\nHalimbawa: Ang iyong kritikal na puna ay nakatulong sa
pagpapabuti ng ulat.";
    e.synonyms = {"essential", "crucial", "urgent"};
    e.antonyms = {"insignificant", "trivial", "unimportant"};
    e.background = "From Greek 'kritikos', able to judge";
    e.usage = "Your critical feedback helped improve the quality of the final report.";
    m_words.append(e);

    e.word = "Credible";
    e.definition = "The journalist gave a credible account of the events that took place.";
    e.translation = "Mapagkakatiwalaan\nKasingkahulugan: believable, reliable,
trustworthy\nKasalungat: unbelievable, unreliable, dubious\nHalimbawa: Nagbigay ang
mamamahayag ng kapanipaniwalang salaysay ng mga pangyayari.";
    e.synonyms = {"believable", "reliable", "trustworthy"};
    e.antonyms = {"unbelievable", "unreliable", "dubious"};
    e.background = "From Latin 'credibilis', believable";
    e.usage = "The journalist gave a credible account of the events that took place.";
    m_words.append(e);

    e.word = "Clarity";
    e.definition = "The clarity of her explanation made the complex concept easy to understand.";
    e.translation = "Kalinawan\nKasingkahulugan: clearness, precision,
transparency\nKasalungat: confusion, ambiguity, vagueness\nHalimbawa: Dahil sa linaw ng
kanyang paliwanag, madali nilang naintindihan ang konsepto.";
    e.synonyms = {"clearness", "precision", "transparency"};
    e.antonyms = {"confusion", "ambiguity", "vagueness"};
    e.background = "From Latin 'claritas', brightness";
    e.usage = "The clarity of her explanation made the complex concept easy to understand.";
    m_words.append(e);

    e.word = "Crowded";
    e.definition = "The subway was crowded during rush hour, making it difficult to move.";
    e.translation = "Matao\nKasingkahulugan: packed, congested, jammed\nKasalungat: empty,
spacious, vacant\nHalimbawa: Matao sa tren tuwing rush hour kaya hirap gumalaw.";
    e.synonyms = {"packed", "congested", "jammed"};
    e.antonyms = {"empty", "spacious", "vacant"};
    e.background = "Common modern English";
    e.usage = "The subway was crowded during rush hour, making it difficult to move.";
    m_words.append(e);

    e.word = "Circular";
    e.definition = "The park had a circular walking path that looped around the lake.";
    e.translation = "Bilog\nKasingkahulugan: round, ring-shaped, annular\nKasalungat: square,
rectangular, angular\nHalimbawa: May paikot na daanan sa parke na pumapalibot sa lawa.";
    e.synonyms = {"round", "ring-shaped", "annular"};
    e.antonyms = {"square", "rectangular", "angular"};
    e.background = "From Latin 'circulus', small ring";
    e.usage = "The park had a circular walking path that looped around the lake.";
    m_words.append(e);

    e.word = "Cuddly";
    e.definition = "The cuddly kitten purred as it curled up in my lap.";
    e.translation = "Malambing\nKasingkahulugan: soft, huggable, snuggly\nKasalungat: rough,
stiff, uninviting\nHalimbawa: Ang malambing na kuting ay umidlip sa aking kandungan.";
    e.synonyms = {"soft", "huggable", "snuggly"};
    e.antonyms = {"rough", "stiff", "uninviting"};
    e.background = "Colloquial usage";
    e.usage = "The cuddly kitten purred as it curled up in my lap.";
    m_words.append(e);

    e.word = "Clamorous";
    e.definition = "The clamorous crowd cheered as the team scored the winning goal.";
    e.translation = "Maingay\nKasingkahulugan: noisy, loud, boisterous\nKasalungat: quiet,
peaceful, subdued\nHalimbawa: Sigawan ang narinig mula sa maingay na karamihan matapos
manalo ang koponan.";
    e.synonyms = {"noisy", "loud", "boisterous"};
    e.antonyms = {"quiet", "peaceful", "subdued"};
    e.background = "From Latin 'clamor', a shout";
    e.usage = "The clamorous crowd cheered as the team scored the winning goal.";
    m_words.append(e);

    e.word = "Cold";
    e.definition = "The cold wind made it feel like winter even though it was still autumn.";
    e.translation = "Malamig\nKasingkahulugan: chilly, frigid, icy\nKasalungat: warm, hot,
toasty\nHalimbawa: Ang malamig na hangin ay nagparamdam ng taglamig kahit taglagas pa.";
    e.synonyms = {"chilly", "frigid", "icy"};
    e.antonyms = {"warm", "hot", "toasty"};
    e.background = "Old English 'cald'";
    e.usage = "The cold wind made it feel like winter even though it was still autumn.";
    m_words.append(e);

    e.word = "Capable";
    e.definition = "She is capable of handling complex tasks under pressure.";
    e.translation = "May Kakayahan\nKasingkahulugan: competent, skilled, able\nKasalungat:
incompetent, incapable, unfit\nHalimbawa: Marunong siyang humawak ng mahihirap na gawain
kahit may pressure.";
    e.synonyms = {"competent", "skilled", "able"};
    e.antonyms = {"incompetent", "incapable", "unfit"};
    e.background = "From Latin 'capax', able to contain";
    e.usage = "She is capable of handling complex tasks under pressure.";
    m_words.append(e);

    e.word = "Captive";
    e.definition = "The animals in the zoo were captive, unable to roam free in the wild.";
    e.translation = "Bilanggo\nKasingkahulugan: imprisoned, confined, enslaved\nKasalungat:
free, liberated, independent\nHalimbawa: Ang mga hayop sa zoo ay bilanggo at hindi makagala
sa kagubatan.";
    e.synonyms = {"imprisoned", "confined", "enslaved"};
    e.antonyms = {"free", "liberated", "independent"};
    e.background = "From Latin 'captivus', taken";
    e.usage = "The animals in the zoo were captive, unable to roam free in the wild.";
    m_words.append(e);
    e.word = "Clear";
    e.definition = "The instructions were clear, and everyone understood what to do.";
    e.translation = "Malinaw\nKasingkahulugan: obvious, evident, transparent\nKasalungat:
unclear, ambiguous, opaque\nHalimbawa: Malinaw ang mga tagubilin kaya lahat ay nakasunod.";
    e.synonyms = {"obvious", "evident", "transparent"};
    e.antonyms = {"unclear", "ambiguous", "opaque"};
    e.background = "From Old English 'cleare'";
    e.usage = "The instructions were clear, and everyone understood what to do.";
    m_words.append(e);

    e.word = "Charitable";
    e.definition = "The charitable organization helps provide food and shelter for the homeless.";
    e.translation = "Mapagbigay\nKasingkahulugan: generous, kind, benevolent\nKasalungat:
selfish, greedy, stingy\nHalimbawa: Ang mapagbigay na organisasyon ay tumutulong sa mga
walang tirahan.";
    e.synonyms = {"generous", "kind", "benevolent"};
    e.antonyms = {"selfish", "greedy", "stingy"};
    e.background = "From Latin 'caritas', charity";
    e.usage = "The charitable organization helps provide food and shelter for the homeless.";
    m_words.append(e);

    e.word = "Content";
    e.definition = "After a long day of work, he felt content sitting on the couch.";
    e.translation = "Kuntento\nKasingkahulugan: satisfied, pleased, happy\nKasalungat:
dissatisfied, unhappy, discontent\nHalimbawa: Pagkatapos ng buong araw ng trabaho, kuntento
siyang naupo sa sofa.";
    e.synonyms = {"satisfied", "pleased", "happy"};
    e.antonyms = {"dissatisfied", "unhappy", "discontent"};
    e.background = "From Latin 'contentus', satisfied";
    e.usage = "After a long day of work, he felt content sitting on the couch.";
    m_words.append(e);

    e.word = "Conserve";
    e.definition = "We need to conserve water during the drought to avoid shortages.";
    e.translation = "Magtipid\nKasingkahulugan: preserve, protect, save\nKasalungat: waste,
squander, deplete\nHalimbawa: Kailangang magtipid ng tubig sa panahon ng tagtuyot.";
    e.synonyms = {"preserve", "protect", "save"};
    e.antonyms = {"waste", "squander", "deplete"};
    e.background = "From Latin 'conservare'";
    e.usage = "We need to conserve water during the drought to avoid shortages.";
    m_words.append(e);

    e.word = "Commendable";
    e.definition = "Her commendable efforts to reduce waste in the office were recognized by
management.";
```

```
    e.translation = "Kapuri-puri\nKasingkahulugan: praiseworthy, admirable,
laudable\nKasalungat: disreputable, dishonorable, blameworthy\nHalimbawa: Ang kanyang
kapuri-puring pagsisikap na bawasan ang basura ay napansin ng pamunuan.";
    e.synonyms = {"praiseworthy", "admirable", "laudable"};
    e.antonyms = {"disreputable", "dishonorable", "blameworthy"};
    e.background = "From Latin 'commendare', to entrust";
    e.usage = "Her commendable efforts to reduce waste in the office were recognized by
management.";
    m_words.append(e);

    e.word = "Composed";
    e.definition = "Despite the chaos around her, she remained composed and kept working.";
    e.translation = "Kalma\nKasingkahulugan: calm, collected, serene\nKasalungat: agitated,
nervous, stressed\nHalimbawa: Sa kabila ng kaguluhan, nanatili siyang kalmado at nagpatuloy sa
trabaho.";
    e.synonyms = {"calm", "collected", "serene"};
    e.antonyms = {"agitated", "nervous", "stressed"};
    e.background = "From Latin 'componere', to put together";
    e.usage = "Despite the chaos around her, she remained composed and kept working.";
    m_words.append(e);

    // Post-process initial words to ensure each entry has a distinct definition and usage.
    // This prevents the UI from showing identical definition and usage.
    for (WordEntry &we : m_words) {
        // Ensure definition exists
        if (we.definition.trimmed().isEmpty()) {
            we.definition = QString("Definition for %1 is not available.").arg(we.word);
        }

        // If usage is empty or identical to definition, generate a short example
        if (we.usage.trimmed().isEmpty() || we.usage.trimmed() == we.definition.trimmed()) {
            const QString def = we.definition.trimmed();
            if (!def.isEmpty()) {
                // Create a different-looking example sentence while reusing the meaning
                we.usage = QString("In context, '%1' can be used like this: %2").arg(we.word, def);
            } else {
                we.usage = QString("Example: '%1' used in a sentence.").arg(we.word);
            }
        }

        // If background is missing this function provides a small placeholder that still looks
informative
        if (we.background.trimmed().isEmpty()) {
            we.background = QString("Background: Etymology or usage information for '%1' is not
provided.").arg(we.word);
        }
    }

}
```

Word_Storage.cpp

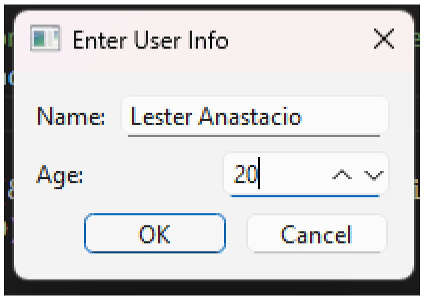
```
#ifndef WORD_STORAGE_H
#define WORD_STORAGE_H

#include <QString>
#include <QStringList>
#include <QVector>
#include <QJsonObject>
#include <QJsonArray>

// Structure to hold data for a single word entry.
struct WordEntry {
    QString word;
    QString definition;
    QStringList synonyms;
    QStringList antonyms;
    QString background;
    QString usage;
    QString translation;

    QJsonObject toJson() const {
        QJsonObject o;
        o["word"] = word;
        o["definition"] = definition;
        QJsonArray syn; for (const auto &s : synonyms) syn.append(s);
        QJsonArray ant; for (const auto &a : antonyms) ant.append(a);
        o["synonyms"] = syn;
        o["antonyms"] = ant;
        o["background"] = background;
        o["usage"] = usage;
        o["translation"] = translation; // save translation
        return o;
    }

    static QStringList jsonArrayToStringList(const QJsonArray &arr) {
        QStringList out;
        out.reserve(arr.size());
        for (const QJsonValue &v : arr) out.append(v.toString());
        return out;
    }

    static WordEntry fromJson(const QJsonObject &o) {
        WordEntry e;
        e.word = o.value("word").toString();
        e.definition = o.value("definition").toString();
        e.synonyms = jsonArrayToStringList(o.value("synonyms").toArray());
        e.antonyms = jsonArrayToStringList(o.value("antonyms").toArray());
        e.background = o.value("background").toString();
        e.usage = o.value("usage").toString();
        e.translation = o.value("translation").toString();
        return e;
    }
};

// Singleton class for managing the dictionary's word storage.
class WordStorage {
public:
    static WordStorage &instance();

    bool load(const QString &path = QString("words.json"));
    bool save(const QString &path = QString());

    void addWord(const WordEntry &entry);
    QVector<WordEntry> allWords() const;
    QVector<WordEntry> wordsForLetter(QChar letter) const;
    bool empty() const { return m_words.isEmpty(); }
    void insertInitialWords();

private:
    WordStorage() = default;
    QVector<WordEntry> m_words;
    QString m_path;
};

#endif // WORD_STORAGE_H
```

Word_Storage.h



```
int main(int argc, char *argv[]) {
    QApplication a(argc, argv);

    // Load existing users (file is users.json in cwd).
    // If a previous user was saved, they will be set as the current user here.
    UserStorage::instance().load("users.json");

    // Show the custom loading screen with title and animated bar.
    LoadingScreen loader;

    QEventLoop loop;
    QObject::connect(&loader, &LoadingScreen::finished, &loop, &QEventLoop::quit);
    loader.start(1[illegible]);
    loop.exec();

    // Show a modal dialog to collect user name and age (user can cancel).
    // Only show the UserDialog if no user is currently set.
    if (UserStorage::instance().currentUser().isEmpty()) {
        UserDialog dlg;
        // The dialog handles saving the new user and setting them as the current
        dlg.exec();
    }
```

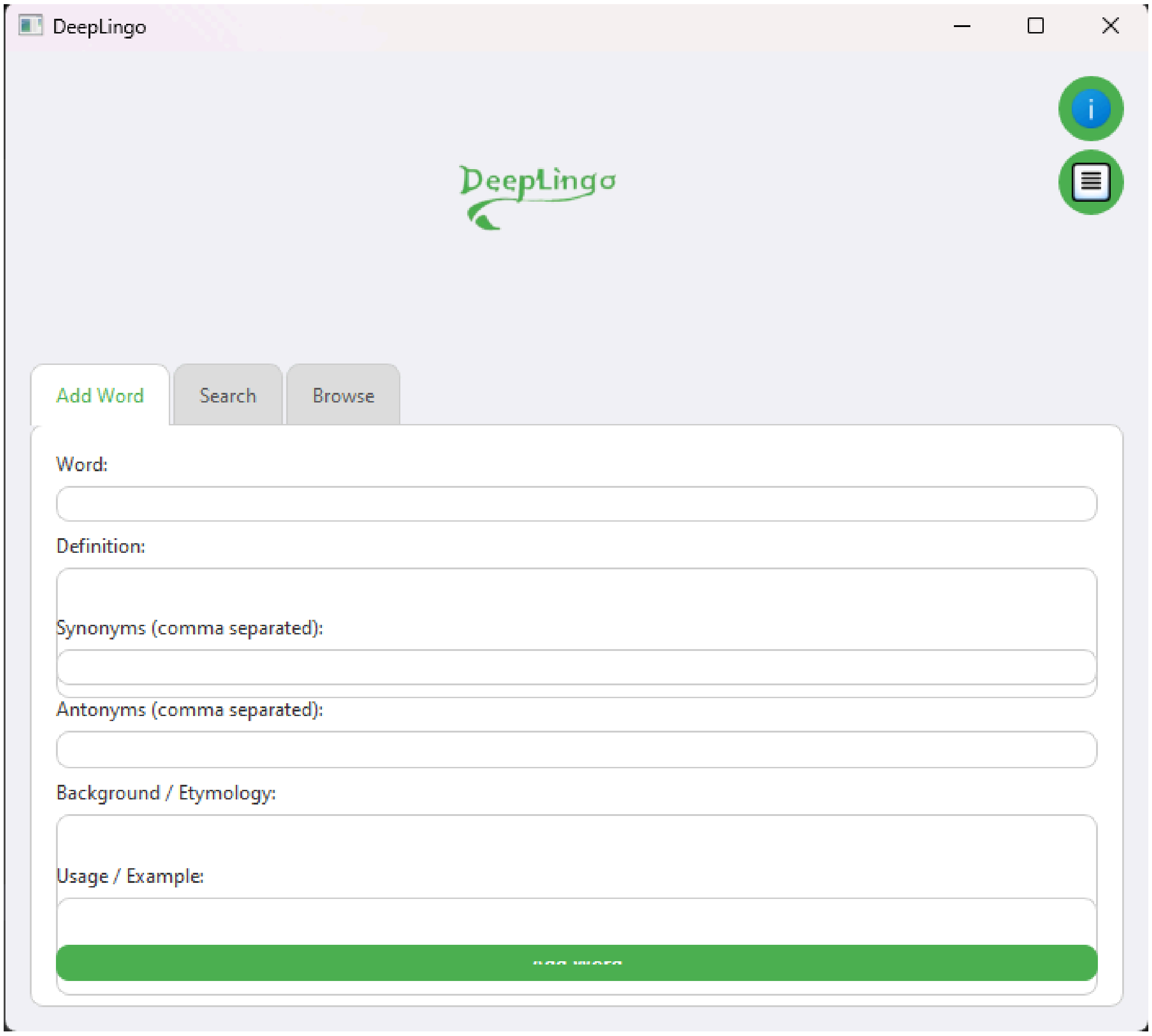
Loading Screen



Name: Lester Anastacio

Age: 20

Upon loading, a user info window will pop and user must type in a name and age



DeepLingo

Word:

Definition:

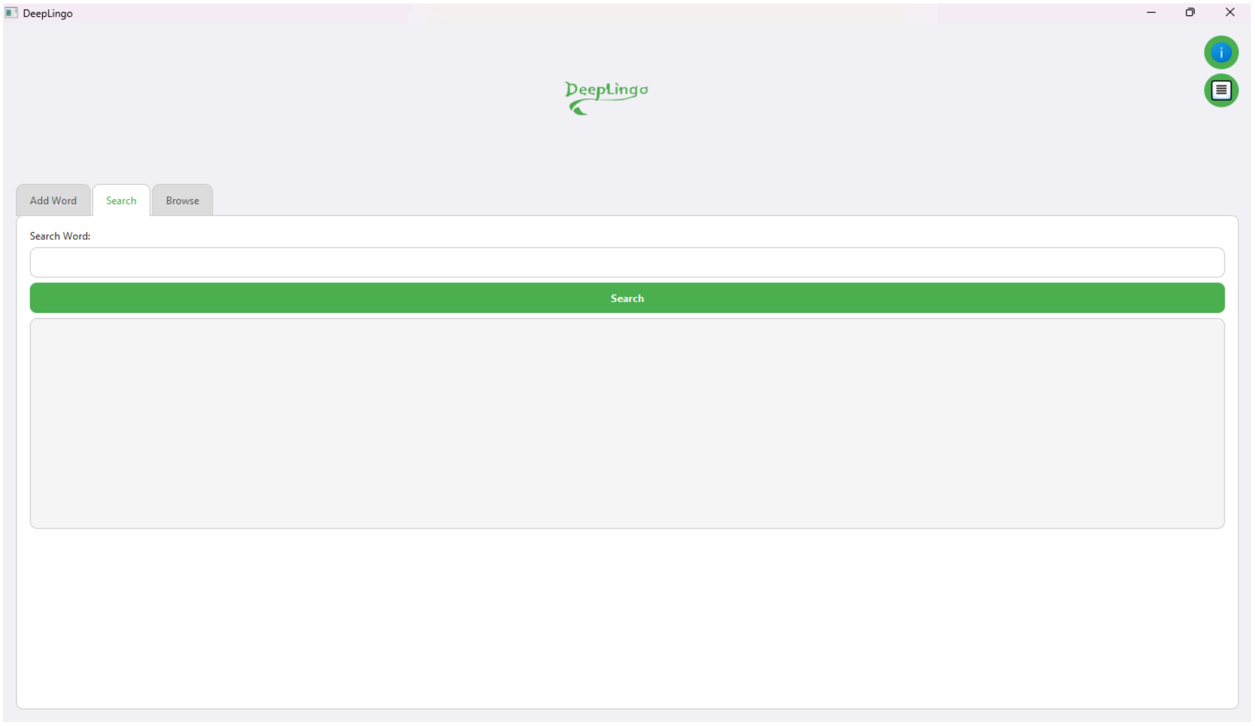
Synonyms (comma separated):

Antonyms (comma separated):

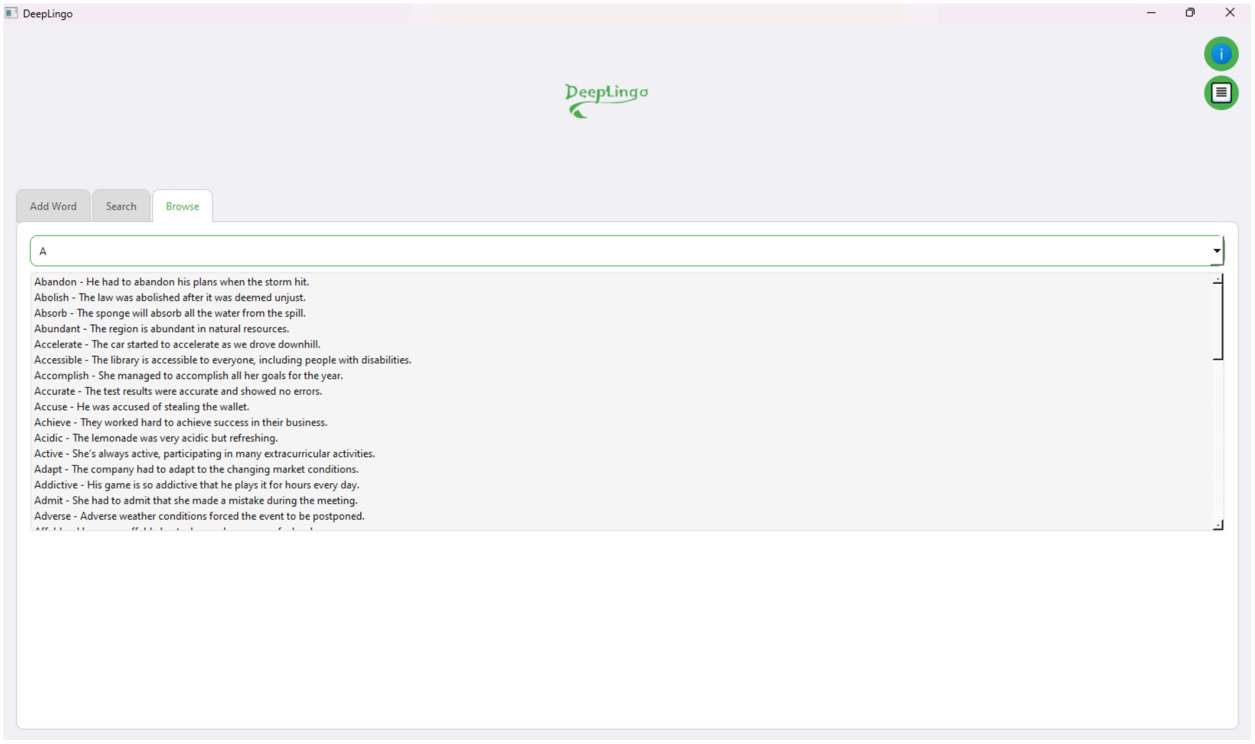
Background / Etymology:

Usage / Example:

Home Screen (Add Word Section)

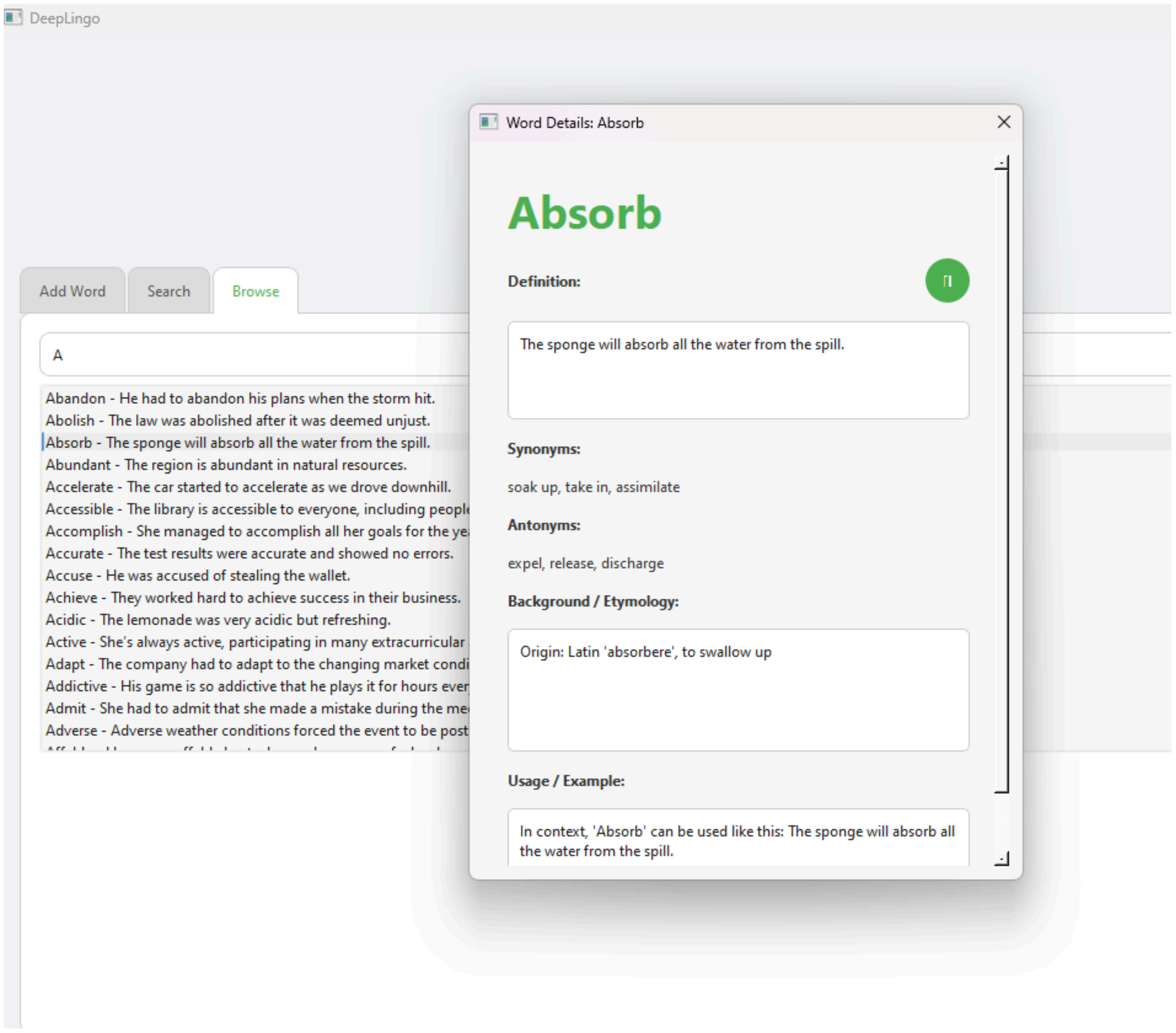


Search Bar Section



```
Abandon - He had to abandon his plans when the storm hit.
Abolish - The law was abolished after it was deemed unjust.
Absorb - The sponge will absorb all the water from the spill.
Abundant - The region is abundant in natural resources.
Accelerate - The car started to accelerate as we drove downhill.
Accessible - The library is accessible to everyone, including people with disabilities.
Accomplish - She managed to accomplish all her goals for the year.
Accurate - The test results were accurate and showed no errors.
Accuse - He was accused of stealing the wallet.
Achieve - They worked hard to achieve success in their business.
Acidic - The lemonade was very acidic but refreshing.
Active - She's always active, participating in many extracurricular activities.
Adapt - The company had to adapt to the changing market conditions.
Addictive - His game is so addictive that he plays it for hours every day.
Admit - She had to admit that she made a mistake during the meeting.
Adverse - Adverse weather conditions forced the event to be postponed.
```

Browse Section



A

Abandon - He had to abandon his plans when the storm hit.
Abolish - The law was abolished after it was deemed unjust.
Absorb - The sponge will absorb all the water from the spill.
Abundant - The region is abundant in natural resources.
Accelerate - The car started to accelerate as we drove downhill.
Accessible - The library is accessible to everyone, including peopl
Accomplish - She managed to accomplish all her goals for the ye
Accurate - The test results were accurate and showed no errors.
Accuse - He was accused of stealing the wallet.
Achieve - They worked hard to achieve success in their business.
Acidic - The lemonade was very acidic but refreshing.
Active - She's always active, participating in many extracurricular
Adapt - The company had to adapt to the changing market condi
Addictive - His game is so addictive that he plays it for hours ever
Admit - She had to admit that she made a mistake during the me
Adverse - Adverse weather conditions forced the event to be post

Absorb

Definition:

The sponge will absorb all the water from the spill.

Synonyms:

soak up, take in, assimilate

Antonyms:

expel, release, discharge

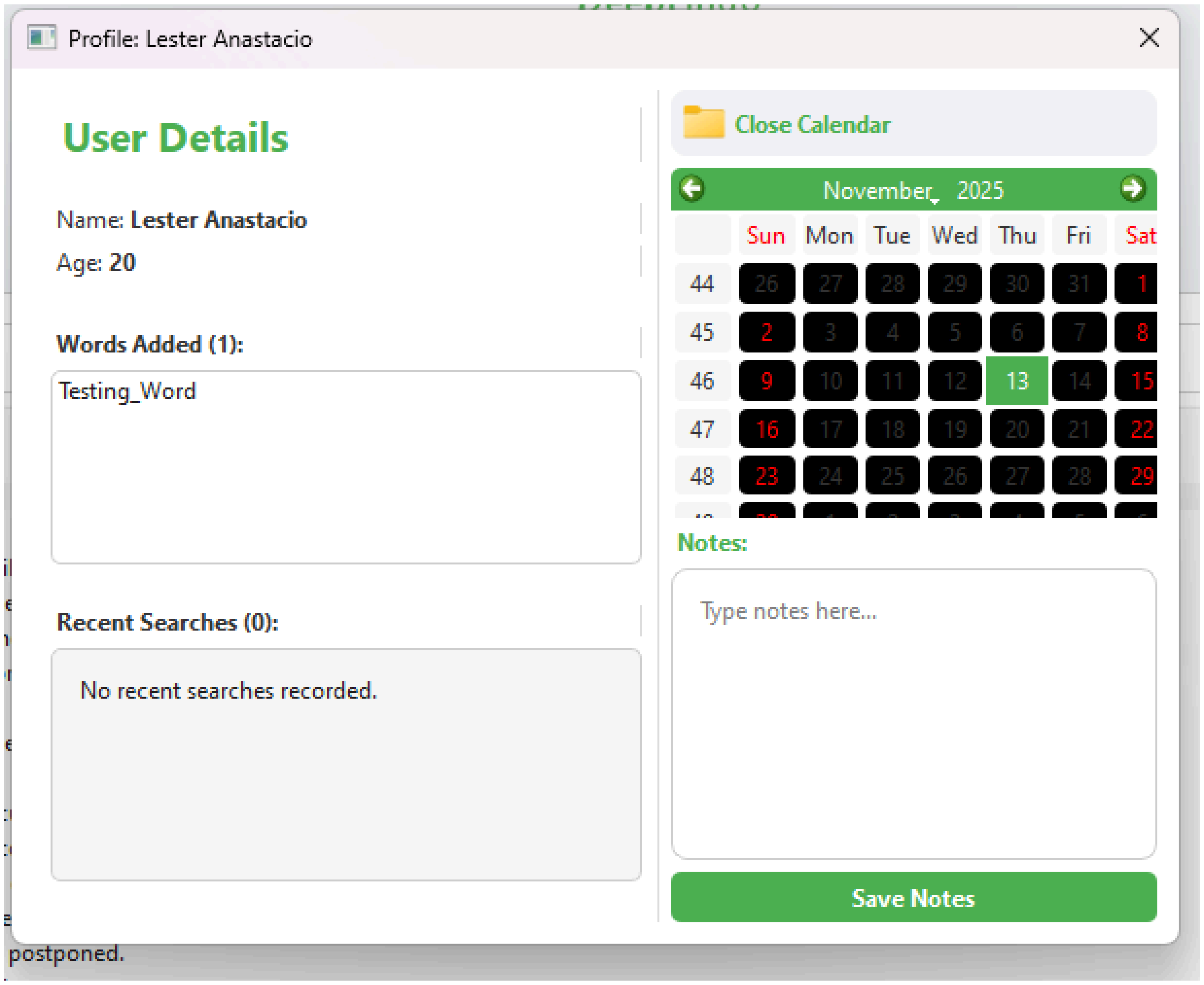
Background / Etymology:

Origin: Latin 'absorbere', to swallow up

Usage / Example:

In context, 'Absorb' can be used like this: The sponge will absorb all the water from the spill.

Word Details Window will pop if user clicked on one of the words



Profile: Lester Anastacio

User Details

Name: **Lester Anastacio**

Age: **20**

Words Added (1):

Testing_Word

Recent Searches (0):

No recent searches recorded.

November 2025

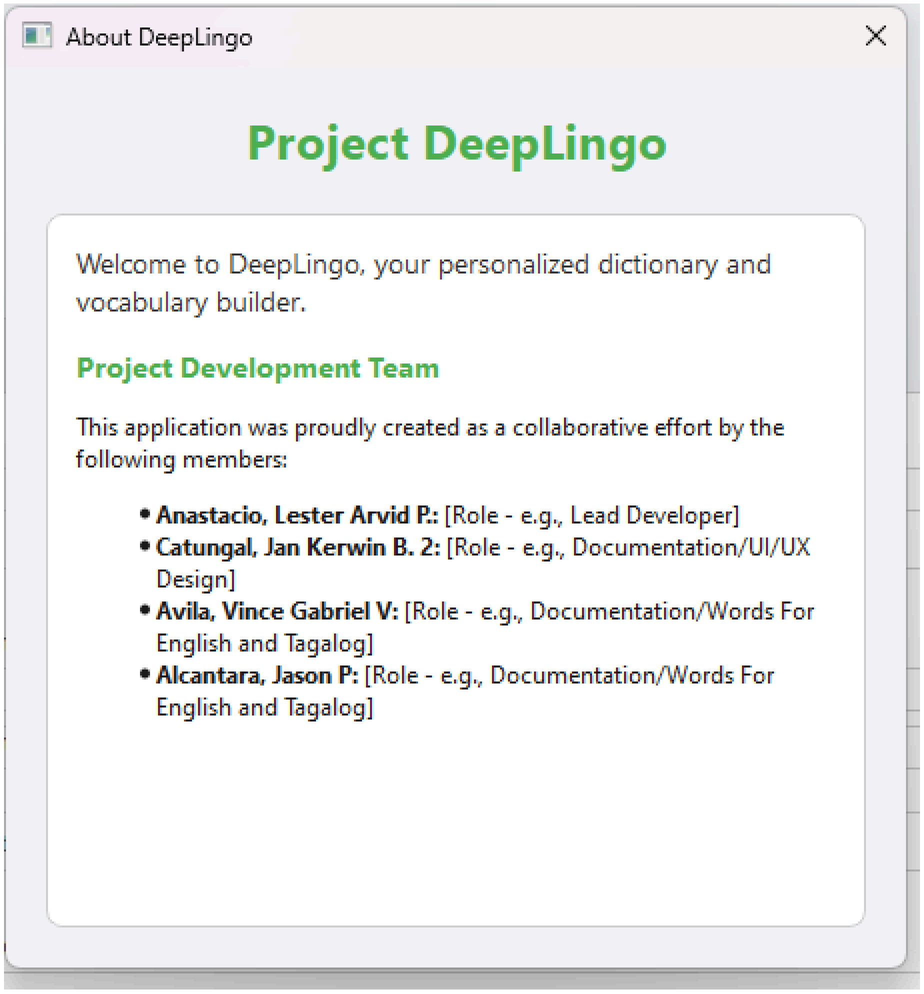
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
44	26	27	28	29	30	31	1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29

Notes:

Type notes here...

postponed.

Profile Window



About DeepLingo

Project DeepLingo

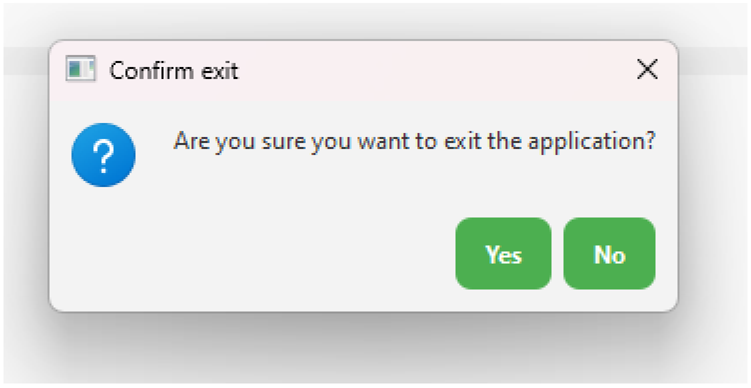
Welcome to DeepLingo, your personalized dictionary and vocabulary builder.

Project Development Team

This application was proudly created as a collaborative effort by the following members:

- **Anastacio, Lester Arvid P.:** [Role - e.g., Lead Developer]
- **Catungal, Jan Kerwin B. 2:** [Role - e.g., Documentation/UI/UX Design]
- **Avila, Vince Gabriel V:** [Role - e.g., Documentation/Words For English and Tagalog]
- **Alcantara, Jason P:** [Role - e.g., Documentation/Words For English and Tagalog]

About Window



Confirm exit

Are you sure you want to exit the application?

Confirmation Window

CONCLUSION

<restate your objectives and conclude your results>

DEMO VIDEO:
https://drive.google.com/drive/folders/1nWIGEQhp0OjXrOaZHgLyJP65wL9gZGjS

REFERENCES

- *QT 6.10. (n.d.). https://doc.qt.io/qt-6/*
- *CMAKE Reference Documentation — CMAKE 4.2.0-RC3 Documentation. (n.d.). https://cmake.org/cmake/help/latest/*
- *C++17 - CpPreference.com. (n.d.). https://en.cppreference.com/w/cpp/17.html*
- *JSON. (n.d.). https://www.json.org/json-en.html*
- *TylerMSFT. (n.d.). Microsoft C/C++ Documentation. Microsoft Learn. https://learn.microsoft.com/en-us/cpp/?view=msvc-170*
- *Qt. (n.d.). qtbase/examples at dev · qt/qtbase. GitHub. https://github.com/qt/qtbase/tree/dev/examples*
- *QT Style Sheets | QT Widgets | QT 6.10.0. (n.d.). https://doc.qt.io/qt-6/stylesheet.html*