**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：物联网工程 班级：物联24x

姓名：Char1esss 学号：U2024xxxxx 实验日期：2025年6月11日

**实验名称：**

哈利波特检索

**实验目的：**

1. 读取7本哈利波特书籍并进行相关处理
2. 检索输入的人名/地名并输出响应结果
3. 选择指定查询结果可显示相关上下文

**实验仪器：**

计算机：Razer Blade 14 2024

CPU：AMD Ryzen 9 7940HS w

内存：32GB 5600MHz DDR5

硬盘：2TB

显卡：Radeon 780M Graphics/NVIDIA RTX 4070 Laptop

操作系统：Windows 11

编译器：MinGW\_x86\_64

集成开发环境：Qt Creator 16.0.1

**实验原理：**

1. 读取查询内容和书目列表
2. 通过章节检测和页码检测获取章节和页码信息，如果行首单词是Chapter则章数+1，如果此行只有数字则页码为此数字。
3. 通过关键词匹配获取查询结果，如果行内包含关键词则返回结果
4. 查询结果显示在文本框内
5. 读取完最后一行关闭文件

**实验内容与步骤：**

// 搜索结果结构体

struct SearchResult {

int fileIndex;

int page;

int chapter;

QString line;

QString fileName;

};

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

signals:

void fileError(const QString& filePath);

void resultFound(const SearchResult& result);

private slots:

void on\_pushButton\_clicked(); // 开始搜索

void on\_pushButton\_2\_clicked(); // 跳转输入行数

void on\_textBrowser\_anchorClicked(const QUrl &url); // 跳转点击行数

void updateProgress(int value); // 更新进度条

void searchFinished(); // 搜索完成处理

void handleResultFound(const SearchResult& result);

private:

void processFile(const QString& filePath, const QString& keyword,int fileIndex); //处理文件

Ui::MainWindow \*ui;

QList<QString> bookFiles\_; // 书籍文件路径列表

QList<SearchResult> searchResults\_; // 搜索结果列表

QRegularExpression chapterRegex; // 章节标题匹配

QFutureWatcher<void> futureWatcher\_; // 任务监视器

QFuture<void> Future; // 任务句柄

};

Mainwindow.h核心部分代码

void on\_pushButton\_clicked()函数接收到“开始搜索”按钮点击信号后首先清空先前输出内容，然后重置进度条，最后遍历书单调用processFil(const QString& filePath, const QString& keyword,int fileIndex)函数进行检索，并在读取完一本书后释放更新进度信号，通过void upgradProgress()更新进度条，全部读取完整后则释放搜索完成信号。

void MainWindow::on\_pushButton\_clicked()

{

// 清空结果并检查输入

searchResults\_.clear();

ui->textBrowser->clear();

QString searchText = ui->lineEdit->text().trimmed();

if (searchText.isEmpty()) return;

// 显示进度条

ui->progressBar->setRange(0, bookFiles\_.size());

ui->progressBar->setValue(0);

// 启动搜索

Future = QtConcurrent::run([=]() {

for (int i = 0; i < bookFiles\_.size(); ++i) {

if (futureWatcher\_.isCanceled()) break;

processFile(bookFiles\_[i], searchText, i);

emit futureWatcher\_.progressValueChanged(i + 1);

}

});

futureWatcher\_.setFuture(Future);

}

void MainWindow::updateProgress(int value)

{

ui->progressBar->setValue(value);

}

void MainWindow::processFile(const QString& filePath, const QString& keyword,int fileIndex)

{

QFile file(filePath);

if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {

emit fileError(filePath);

return;

}

QTextStream in(&file);

int currentPage = 0;

int currentChapter = 0;

while (!in.atEnd()) {

QString line = in.readLine().trimmed();

if (line.isEmpty()) continue;

// 章节检测

if (chapterRegex.match(line).hasMatch()) {

currentChapter++;

continue;

}

// 页码检测

bool ok;

int num = line.toInt(&ok);

if (ok && num > 0 && line == QString::number(num)) {

currentPage = num;

continue;

}

// 关键字匹配

if (line.contains(keyword, Qt::CaseInsensitive)) {

emit resultFound({

fileIndex,

currentPage,

currentChapter,

line,

QFileInfo(filePath).fileName()

});

}

}

file.close();

}

void processFile(const QString& filePath, const QString& keyword,int fileIndex)函数被调用后首先打开对应书目文件，读取错误则释放错误信号并返回。读取成功则逐行读取，若该行为空则跳过，若该行首单词为Chapter则章节号增加，若该行仅有数字则更新页码，然后检测该行存在关键词则释放找到信号，处理完最后一行后关闭文件。

void MainWindow::handleResultFound(const SearchResult& result)

{

searchResults\_.append(result);

int index = searchResults\_.size();

QString displayText = QString("<a href='item:%1'>%2</a> &nbsp;&nbsp; %3 nbsp;&nbsp; Page %4 &nbsp;&nbsp; Chapter %5 &nbsp;&nbsp; %6")

.arg(index)

.arg(index)

.arg(result.fileName)

.arg(result.page)

.arg(result.chapter)

.arg(result.line.left(30) + "...");

ui->textBrowser->append(displayText);

}

void handleResultFound(const SearchResult& result)函数在接收到找到信号后将结果显示在文本框上。

void MainWindow::searchFinished()

{

if (!searchResults\_.isEmpty()) {

ui->statusbar->showMessage(QString("Found %1 results").arg(searchResults\_.size()));

} else {

ui->statusbar->showMessage("No results found.");

}

}

void searchFinished()函数在接收到搜索完成信号后显示结果统计。

void MainWindow::on\_pushButton\_2\_clicked()

{

bool ok;

int input = ui->lineEdit\_2->text().toInt(&ok);

if (!ok || input < 1 || input > searchResults\_.size()) {

QMessageBox::warning(this, "Error", "Invalid entry number");

return;

}

ui->textBrowser\_2->setText(searchResults\_[input - 1].line);

}

void on\_pushButton\_2\_clicked()在用户输入指定序号并点击“查看详情按钮后”会在文本框显示该行内容。

void MainWindow::on\_textBrowser\_anchorClicked(const QUrl &url)

{

if (url.scheme() == "item") {

bool ok;

int index = url.path().toInt(&ok);

if (ok && index >= 1 && index <= searchResults\_.size()) {

ui->textBrowser\_2->setText(searchResults\_[index - 1].line);

}

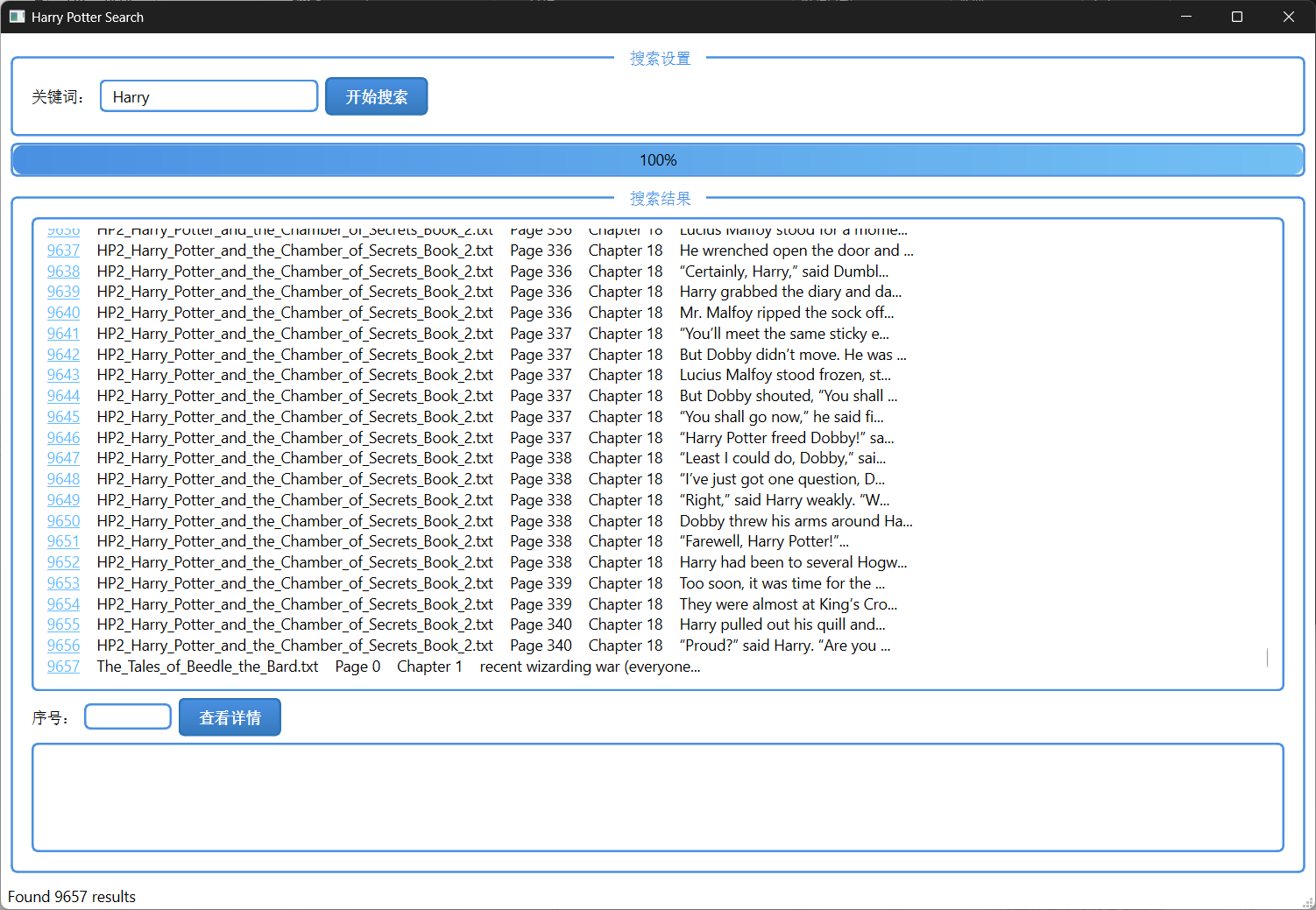
}

void on\_textBrowser\_anchorClicked(const QUrl &url)函数在用户点击序号后再文本框显示该行内容。

**实验结果与分析：**

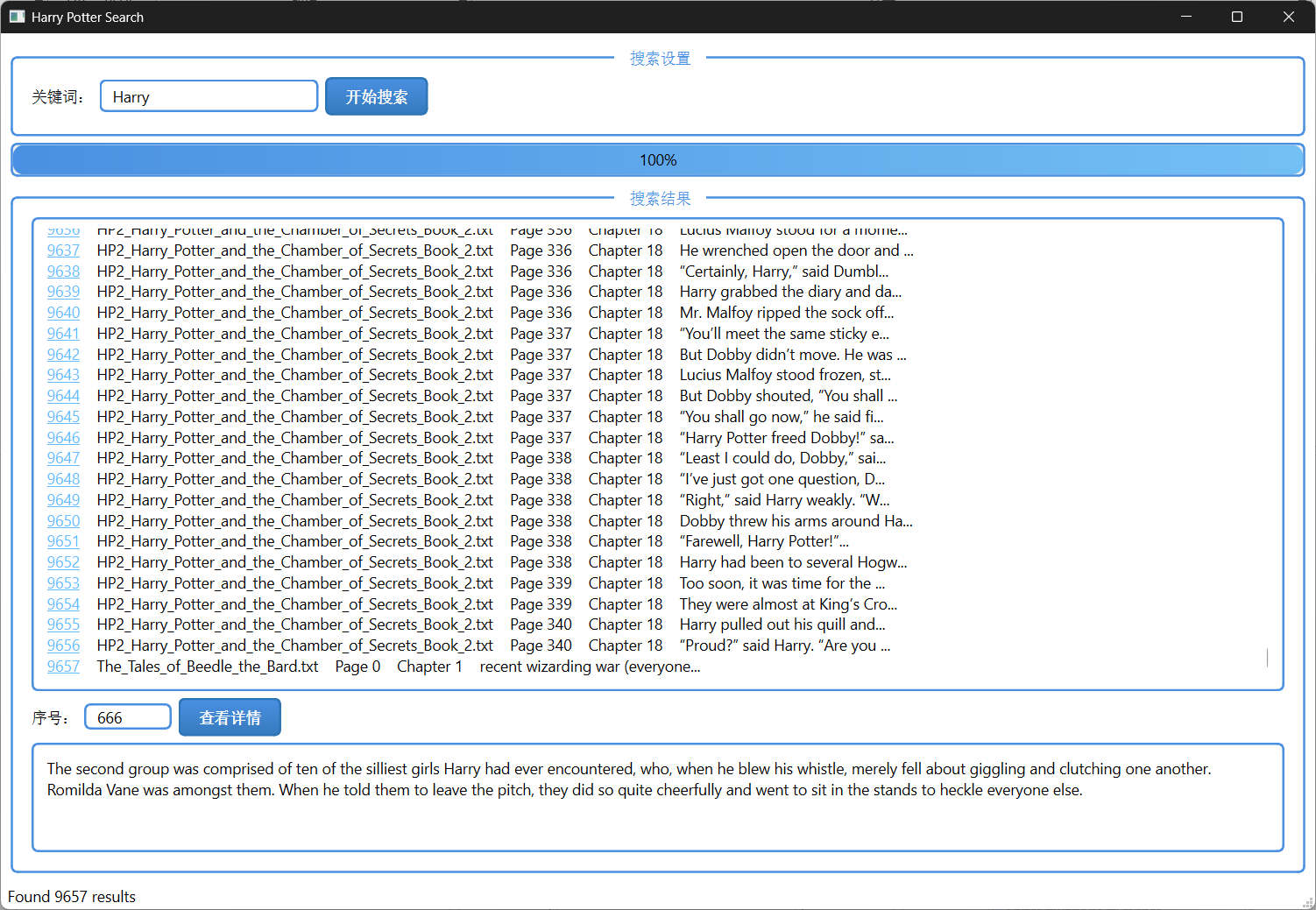
测试用例1：在关键词输入框内输入Harry，点击开始搜索。

运行截图1：显示查询到的9657个结果。

****

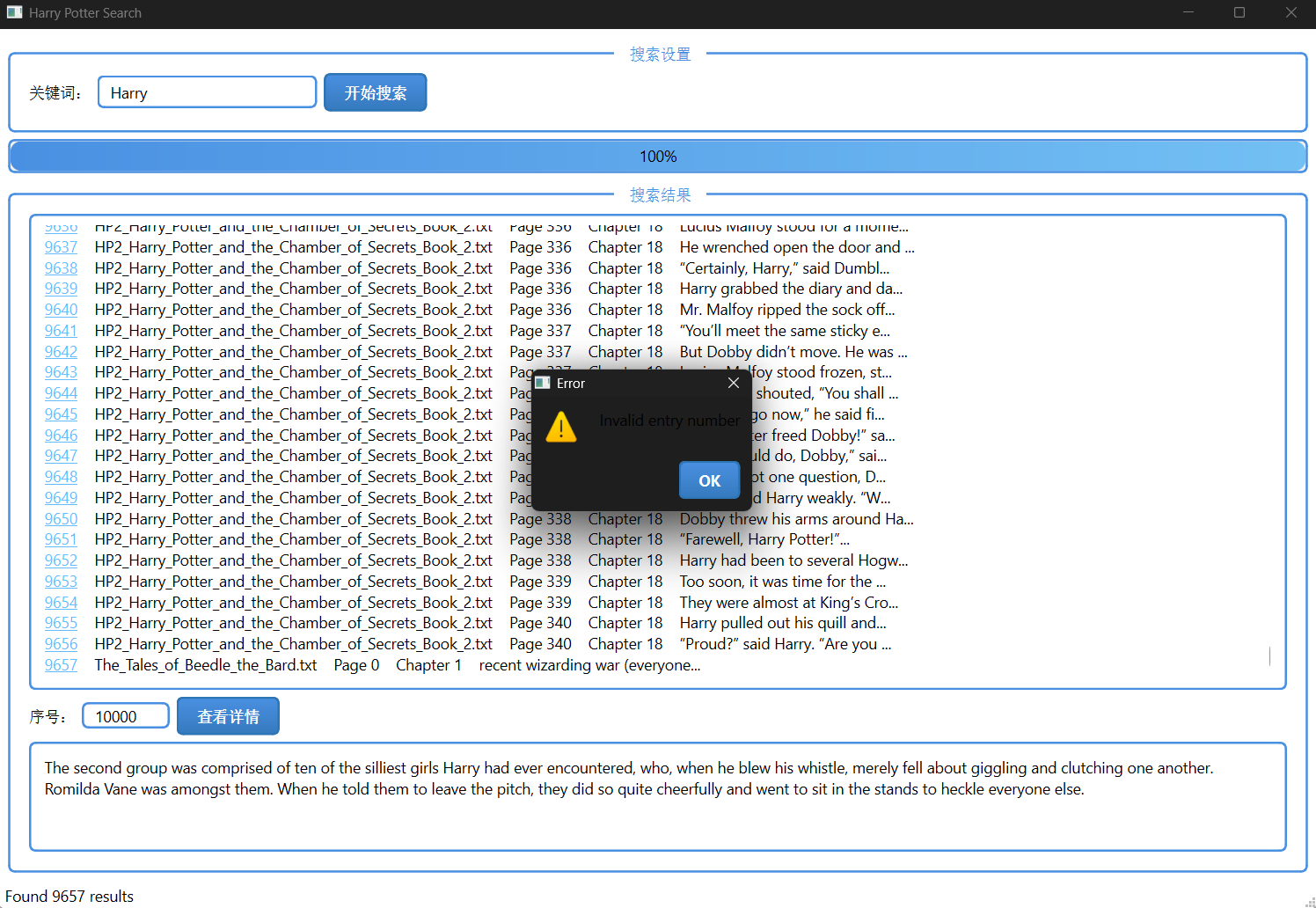
测试用例2：在序号框输入666，点击查看详情

运行截图2：显示该搜索结果内容。



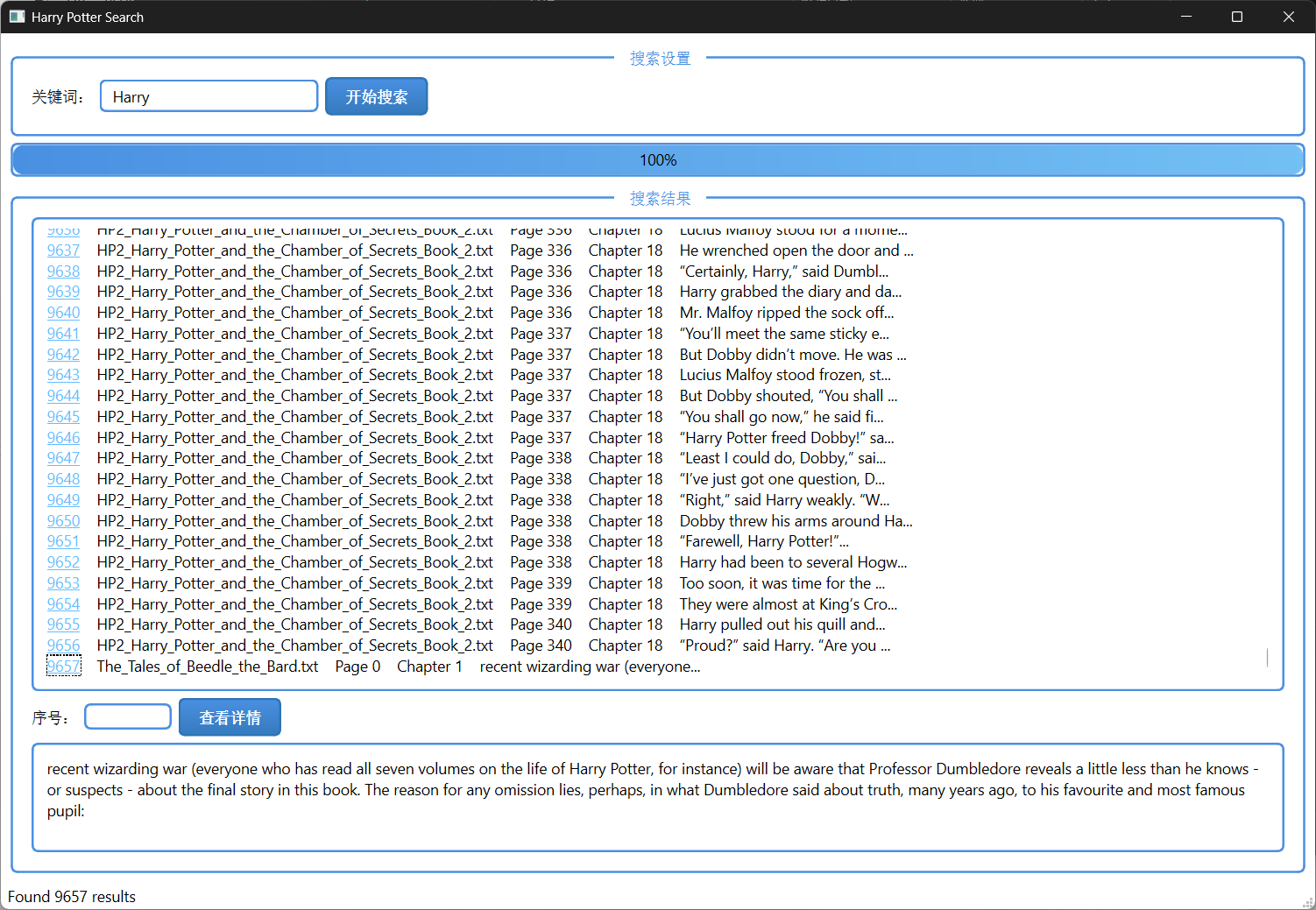
测试用例3：在序号框输入10000，点击查看详情

运行截图3：显示该输入序号无效。



测试用例4：点击序号9657

运行截图4：显示该搜索结果内容。



**实验结论与感想**

非常好的实验，使我的大脑旋转。