 [从俄语翻译成中文(简体) - www.onlinedoctranslator.com](https://www.onlinedoctranslator.com/zh-CN/?utm_source=onlinedoctranslator&utm_medium=docx&utm_campaign=attribution)

俄罗斯联邦教育部莫斯科国立电子技术学院（技术大学）信息与计算机系统软件系

经主管批准。 IPOVS系，技术科学博士，教授。加加林娜 L.G. “\_\_”\_\_\_\_\_\_\_\_\_2024

实验室工作3号

平台选择和项目分解。使用 Git 版本控制系统的基础知识。

系主任、博士、副教授\_\_\_\_\_\_\_\_

表演者，学生 gr。 PIN-36 沃尔科夫 R.S.戈尔班 V.G.  
卡什帕罗夫 G.I.  
卡斯纳什 A.A.

莫斯科，2024 年

### 1. 实施汽车服务自动化计划的技术清单

为了实施汽车服务自动化计划，选择了以下技术：

1. **编程语言**：C++
2. **数据库管理系统（DBMS）**： PostgreSQL
3. **开发环境（IDE）**：Visual Studio 或 CLion
4. **GUI开发框架**：Qt
5. **用于与数据库交互的库**: libpqxx（PostgreSQL 的 C++ 客户端库）
6. **版本控制系统**: git
7. **测试工具**：用于单元测试的 Google Test 或 Catch2

### 2. 选择技术和易用性的理由

1. **C++**:
   * **高性能**。汽车服务可以处理大量与客户、汽车和服务相关的数据和请求，这需要快速处理。
   * **广泛的内存管理选项**。 C++ 允许您更有效地使用内存和系统资源。
   * **大型图书馆生态系统**。 C++ 支持许多库，包括用于处理数据库、创建用户界面和多线程编程的库。
2. **PostgreSQL**:
   * **可靠性和可扩展性**。 PostgreSQL 是最稳定、最可靠的 DBMS 之一，支持事务和数据完整性，这对于汽车服务中心的会计非常重要。
   * **广泛的 SQL 功能**。 PostgreSQL支持复杂的查询、索引和外键，方便处理大量数据。
   * **扩展支持**。 PostgreSQL 有充足的机会通过插件和模块扩展功能。
3. **Qt**:
   * **多平台**。 Qt 允许您开发适用于 Windows、Linux 和 macOS 的程序，只需对代码进行最少的更改。
   * **直观的界面**。 Qt 提供了创建用户友好且有吸引力的用户界面的功能，这对于最终用户来说非常重要。
   * **用于处理图形、文件和网络的强大库**。
4. **libpqxx**:
   * **与 PostgreSQL 直接交互**。 libpqxx 提供了一个在 C++ 中使用 PostgreSQL 的接口，没有不必要的复杂性。
   * **支持所有 PostgreSQL 功能**。该库支持处理事务、复杂查询和准备好的表达式。
5. **git**:
   * **版本控制**。对于团队开发和跟踪代码更改，需要版本控制系统。 Git 允许您有效地管理变更历史记录，从而提高团队生产力。
6. **谷歌测试/Catch2**:
   * **单元测试**。必须保证代码质量并及时发现错误。 Google Test 和 Catch2 是用于在 C++ 中创建单元测试的流行库。

### 3、开发系统分解及模块说明

该系统将由以下主要模块组成：

1. **客户管理模块**:
   * 客户信息的登记和存储。
   * 编辑和删除客户数据。
   * 按参数（姓名、电话号码、订单历史记录）搜索客户。
2. **车辆控制模块**:
   * 维护汽车数据库（品牌、型号、制造年份、VIN 代码）。
   * 将汽车与客户联系起来。
   * 记录车辆维修和保养历史。
3. **服务和订单核算模块**:
   * 创建和管理服务订单。
   * 汽车服务管理（维修、诊断、备件更换）。
   * 存储有关已执行工作、服务时间和成本的信息。
4. **员工管理模块**:
   * 管理员工信息（姓名、职位、联系人）。
   * 将订单和服务与负责实施的员工联系起来。
5. **仓库模块**:
   * 备件和消耗品的核算。
   * 订单完成后自动从仓库核销。
   * 关于需要补充库存的通知。
6. **报告模块**:
   * 生成有关订单、服务、收入的报告。
   * 有关客户、汽车和员工的统计数据。
   * 能够以 PDF 或 Excel 格式导出报告。
7. **安全和访问控制模块**:
   * 用户的授权和认证。
   * 访问权限的区分（管理员、经理、机械师）。
   * 记录用户操作。

**4. 表明其优先级的完整任务列表：**

### 高优先级：

1. **设置开发环境**:
   * 工具的安装和配置：C++、PostgreSQL、Qt、libpqxx、IDE（Visual Studio 或 CLion）、Git。
   * 版本控制系统配置（创建存储库、设置 CI/CD）。
2. **数据库开发**:
   * **数据库结构设计**：创建用于存储客户、汽车、订单、服务、员工、备件数据的表。
   * **数据模式实施**：使用SQL创建表、表之间的关系以及数据完整性约束（例如外键）。
3. **通过 libpqxx 与 PostgreSQL 集成**:
   * **连接设置**：建立与数据库的连接并处理请求的模块的实现。
   * **测试基本操作**：执行CRUD操作（创建、读取、更新、删除）来测试数据库。
4. **客户管理模块开发**:
   * **为客户进行CRUD操作**：创建、编辑、删除、搜索客户。
   * **与客户交互的界面**：用于处理客户数据的便捷 UI。
5. **汽车控制模块的开发**:
   * **汽车的 CRUD 操作**：车辆数据的存储和管理，包括与客户的链接。
   * **与订单集成**：将汽车与特定订单和服务联系起来的能力。
6. **开发会计订单和服务模块**:
   * **订单的创建和管理**：创建新订单、分配服务、管理订单状态（例如“待处理”、“进行中”、“已完成”）的功能。
   * **服务成本的会计处理**：计算和存储服务成本数据。

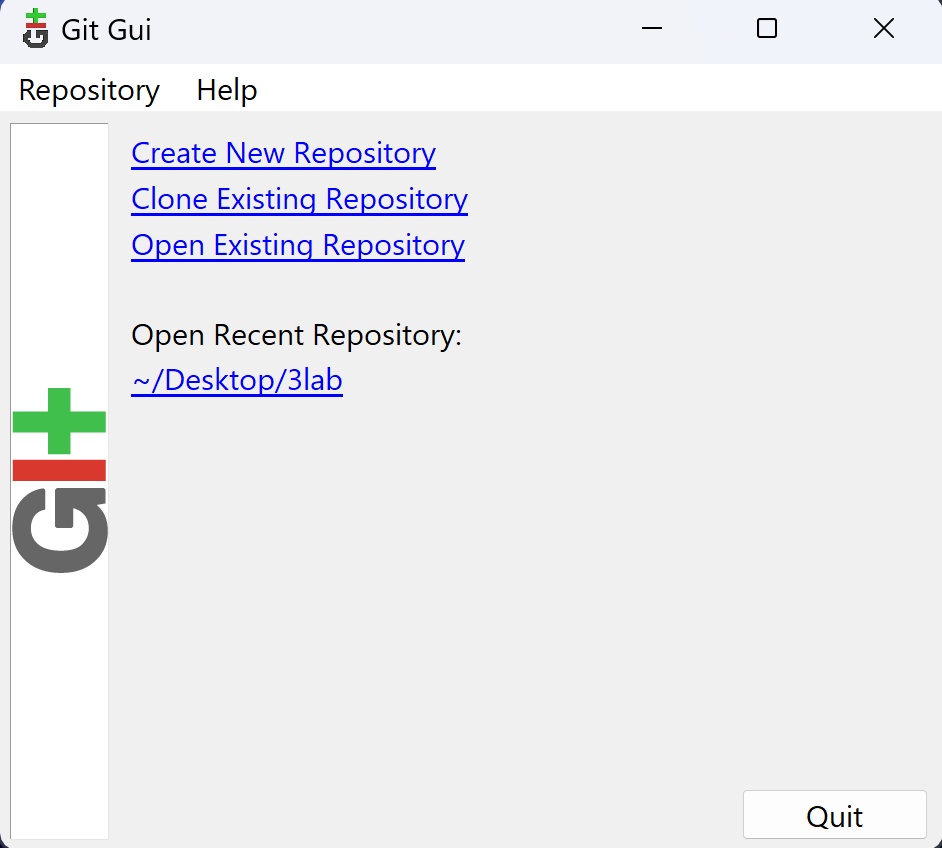
### 中等优先级：

1. **员工管理模块**:
   * **员工的CRUD操作**：创建和编辑员工档案（机械师、管理员）。
   * **将员工与订单关联起来**：任命负责提供服务的员工。
2. **仓库模块**:
   * **库存核算**：存储备件、消耗品的数据。
   * **存货**：订单完成后从仓库注销，并自动通知低余额。
   * **订购新耗材**：用于订购和管理备件接收的界面。
3. **创建用户界面 (UI)**:
   * **基本屏幕**：为所有系统模块开发直观的图形界面。
   * **导航**：实现各个部分（客户、汽车、订单、仓库）之间的导航。
   * **搜索和过滤数据**：通过关键参数搜索客户、订单、车辆的便捷功能。

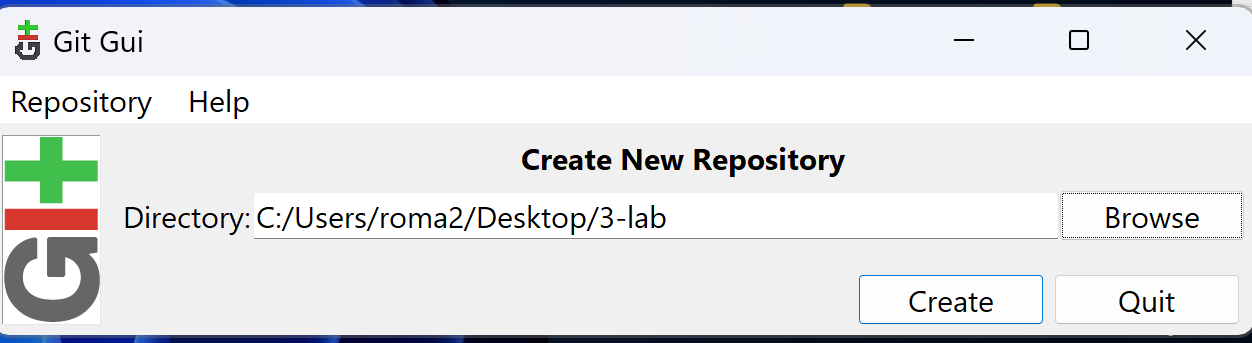
### ****低优先级：****

1. **报告模块**:
   * 生成有关订单、服务、员工的报告。
   * 导出为 PDF/Excel。
   * 统计和分析（收入、受欢迎的服务）。
2. **安全和访问控制模块**:
   * 用户身份验证（登录名/密码）。
   * 按角色（管理员、机械师）区分访问权限。
   * 记录用户操作（谁更改了什么以及何时更改）。
3. **测试与优化**:
   * 单元测试和集成测试。
   * 负载测试以检查性能。
   * 优化数据库查询和代码。
4. **文档**:
   * 有关系统架构和结构的技术文档。
   * 开发人员和用户指南。
   * 用于解决常见问题的常见问题解答。

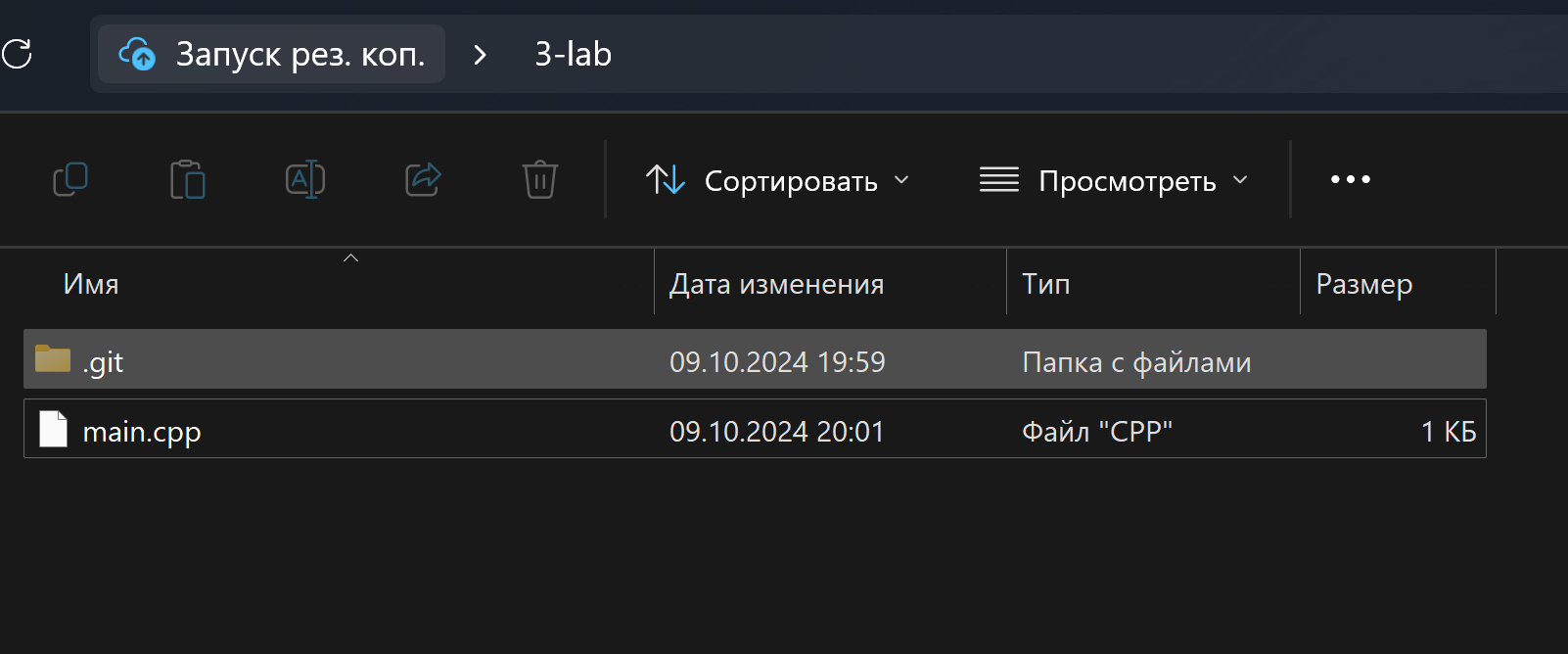
**使用 Git**



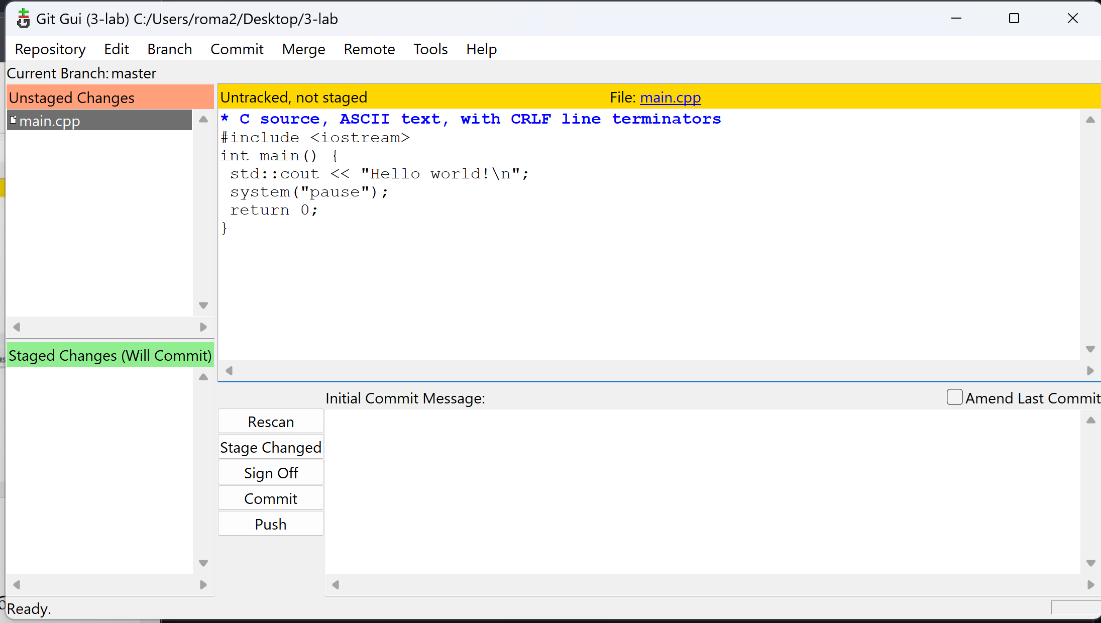
绘画1。启动 Git Gui。



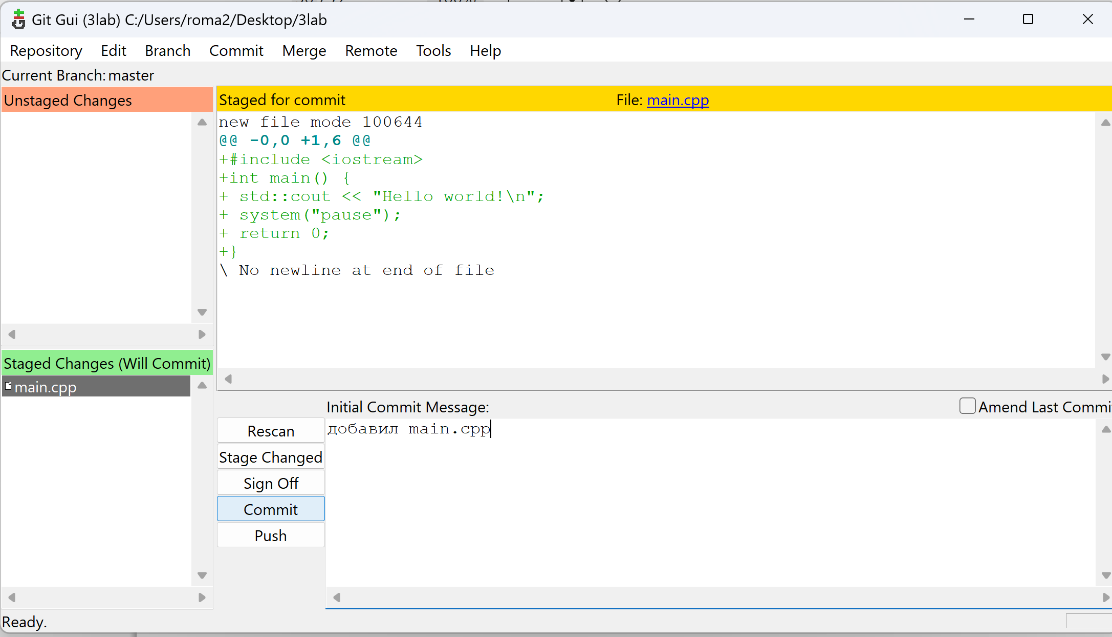
绘画2。为项目创建一个新的存储库。



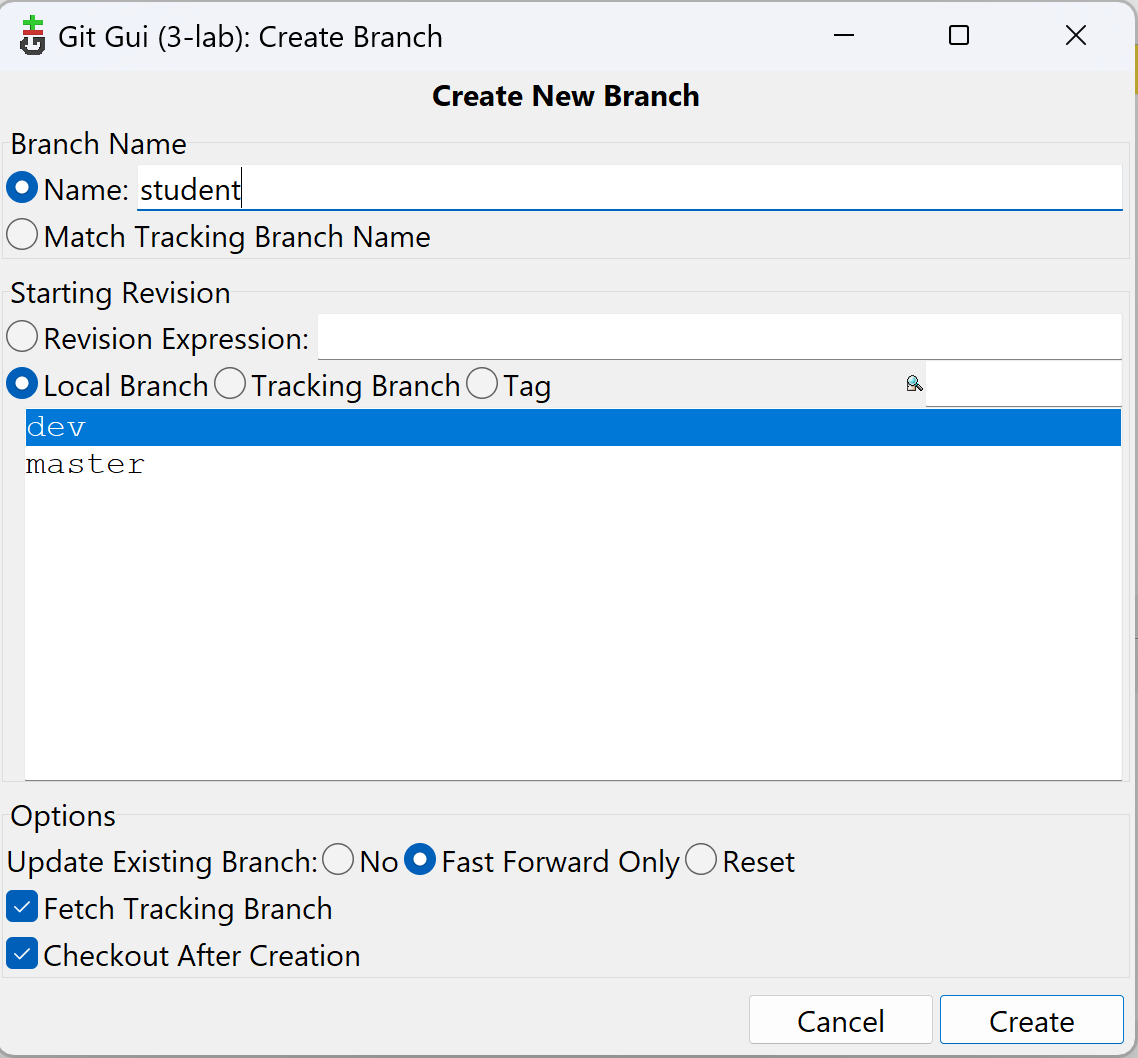
绘画3。将 main.cpp 文件添加到存储库文件夹中。



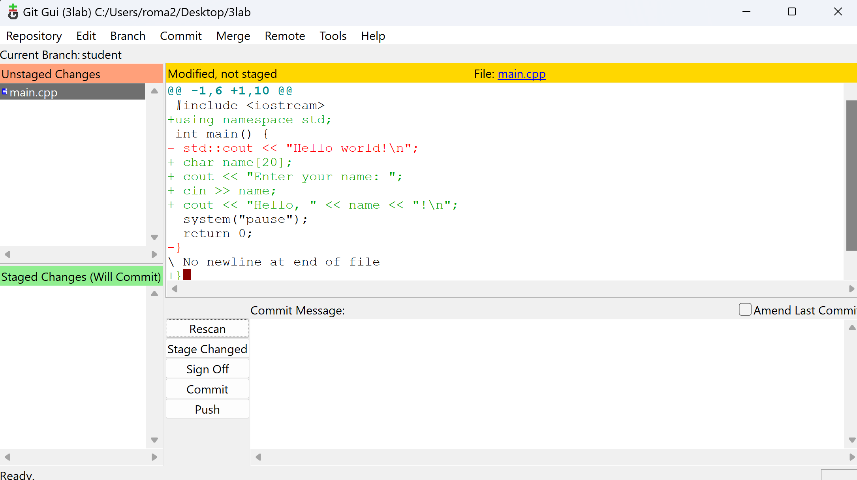
绘画4。单击“重新扫描”后的 Git Gui。



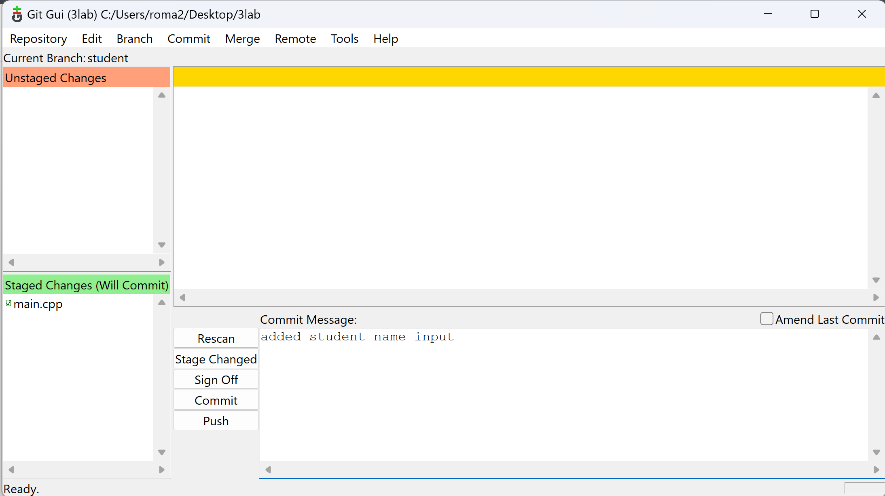
绘画5。在 master 分支上提交 main.cpp 文件。



绘画6。添加分支 dev、student。



绘画7。更改 main.cpp 文件并提交

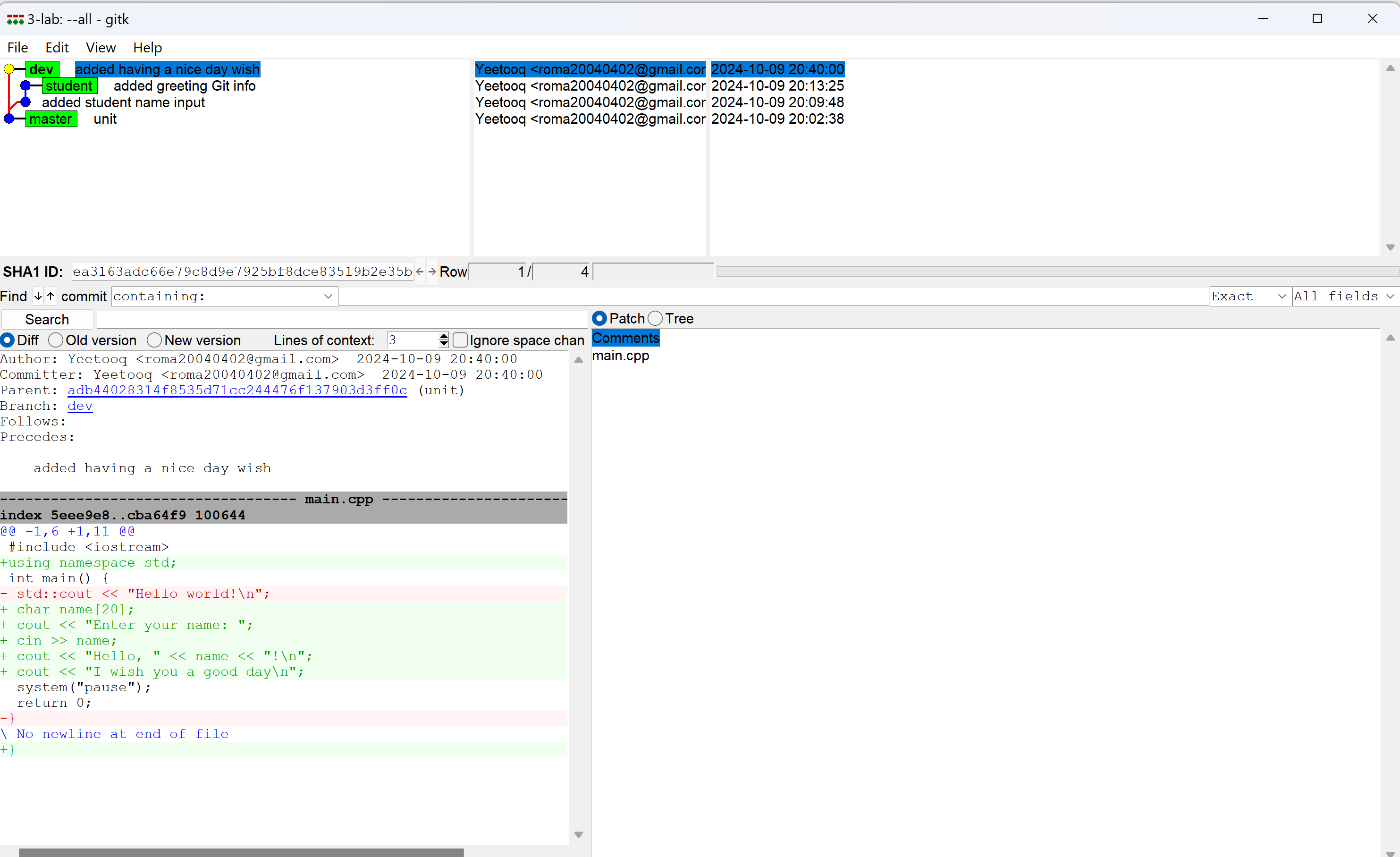


绘画8。在学生分支上提交修改后的main.cpp。

。

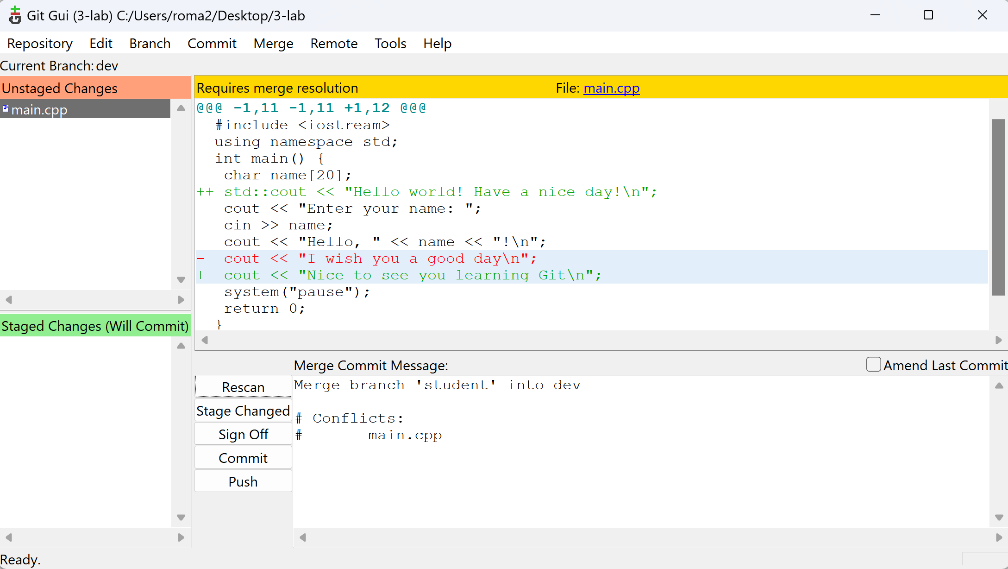
接下来，通过 Branch – Checkout，选择 dev 分支，并进行与 Student 分支相同的工作。

我们对 master 分支执行相同的工作，并通过存储库显示提交历史记录 – 可视化所有分支历史记录：

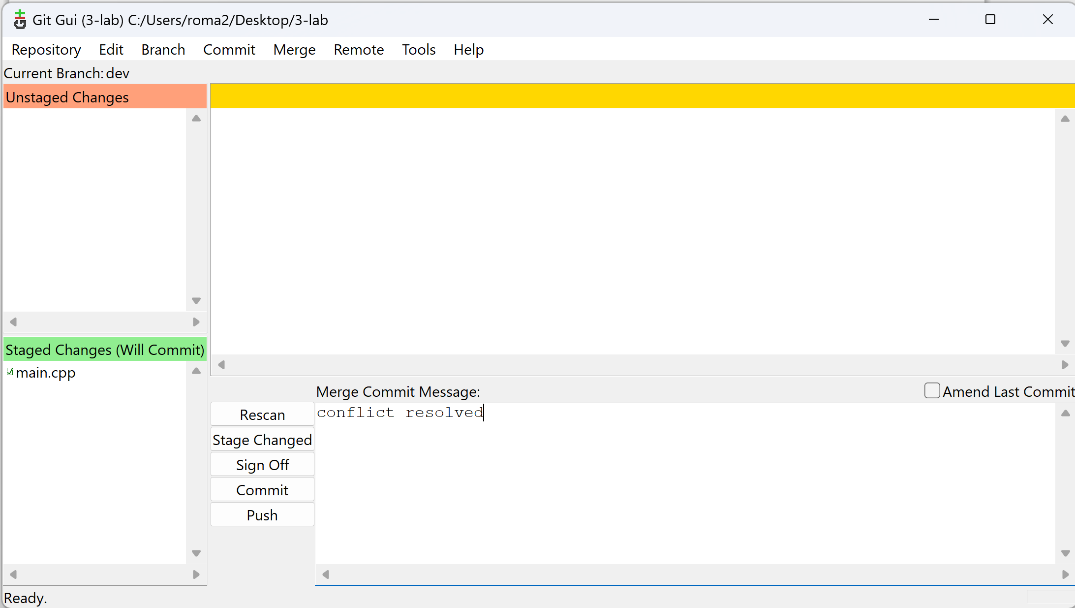


绘画9。提交历史。

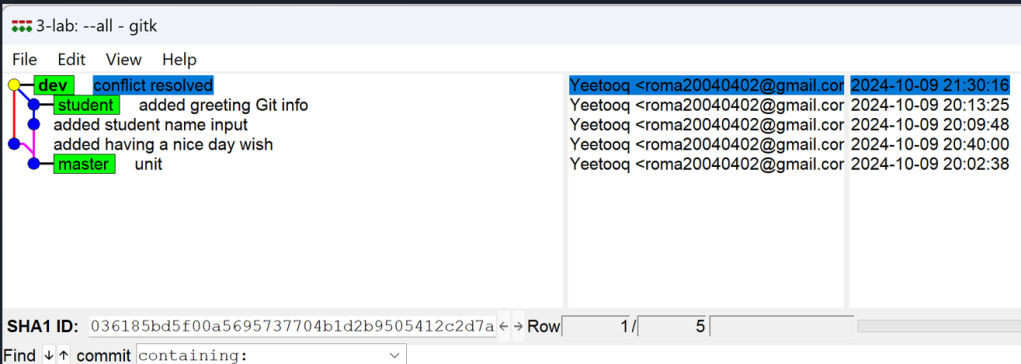
我们更改dev分支中的main.cpp文件，并将dev与student合并：



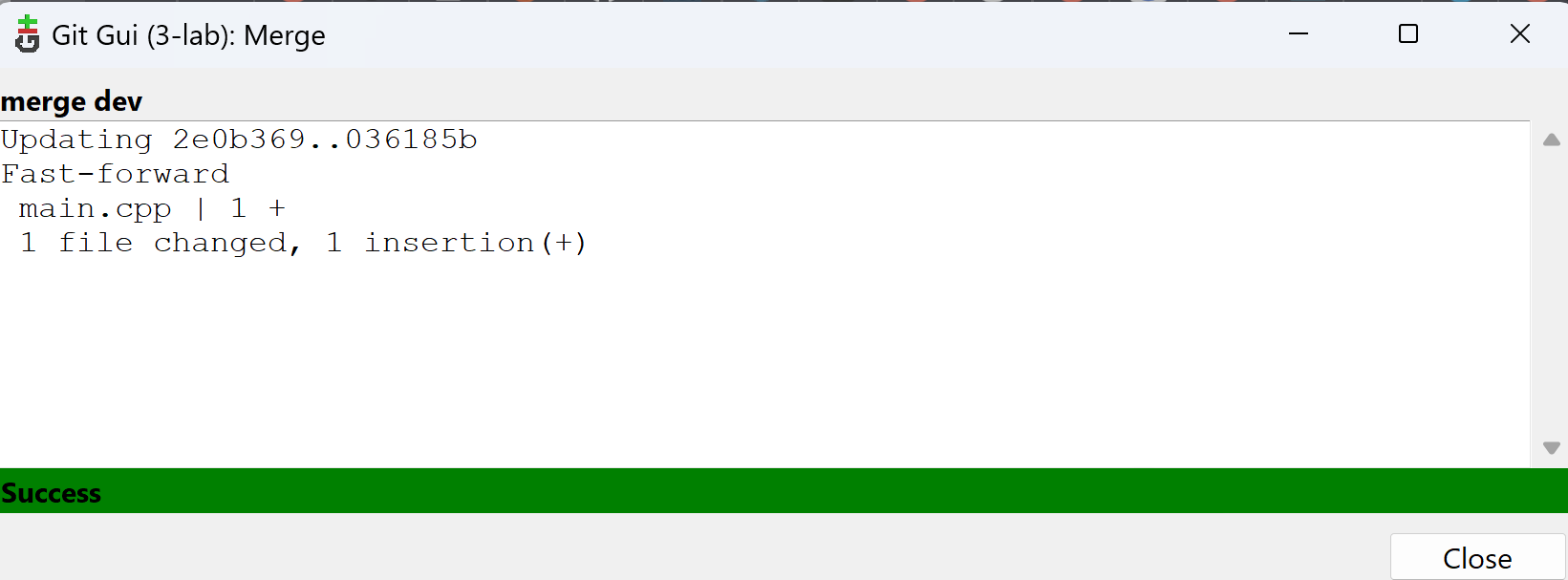
绘画10。更改了 dev 分支上的 main.cpp。

––––

绘画11。将修改后的main.cpp提交到dev分支上。



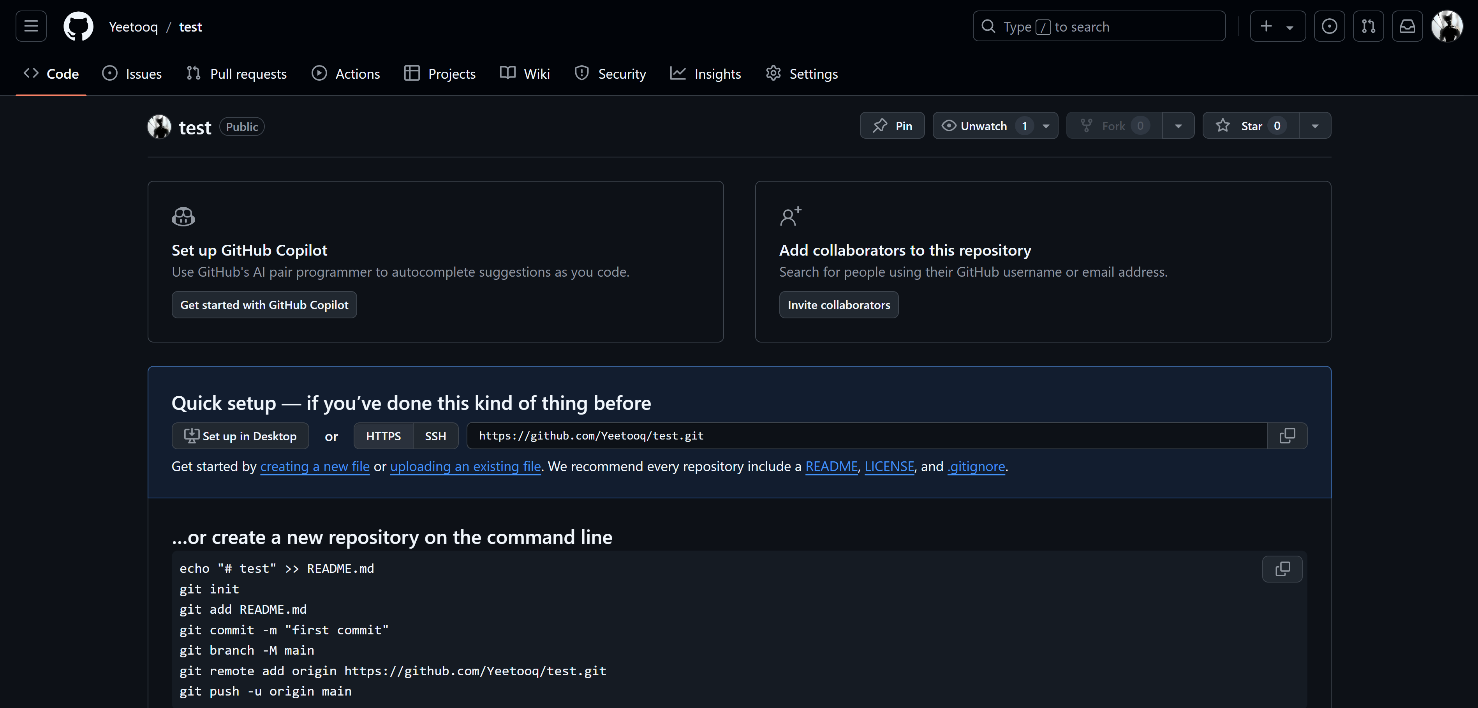
绘画12。更改开发后提交历史记录。



绘画13。学生与开发人员的成功合并。

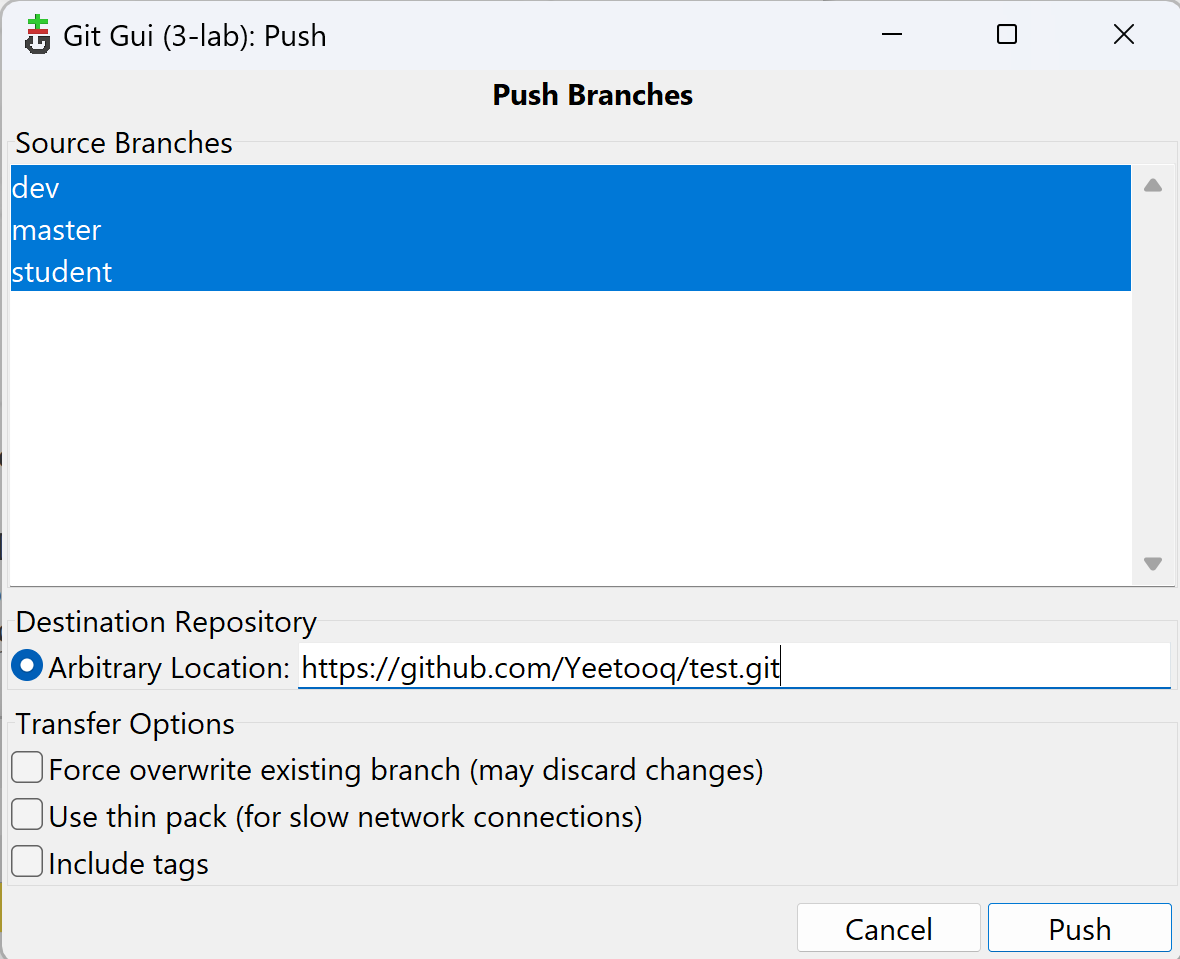
接下来，以此类推，我们将dev分支合并到master分支中。

创建远程存储库：



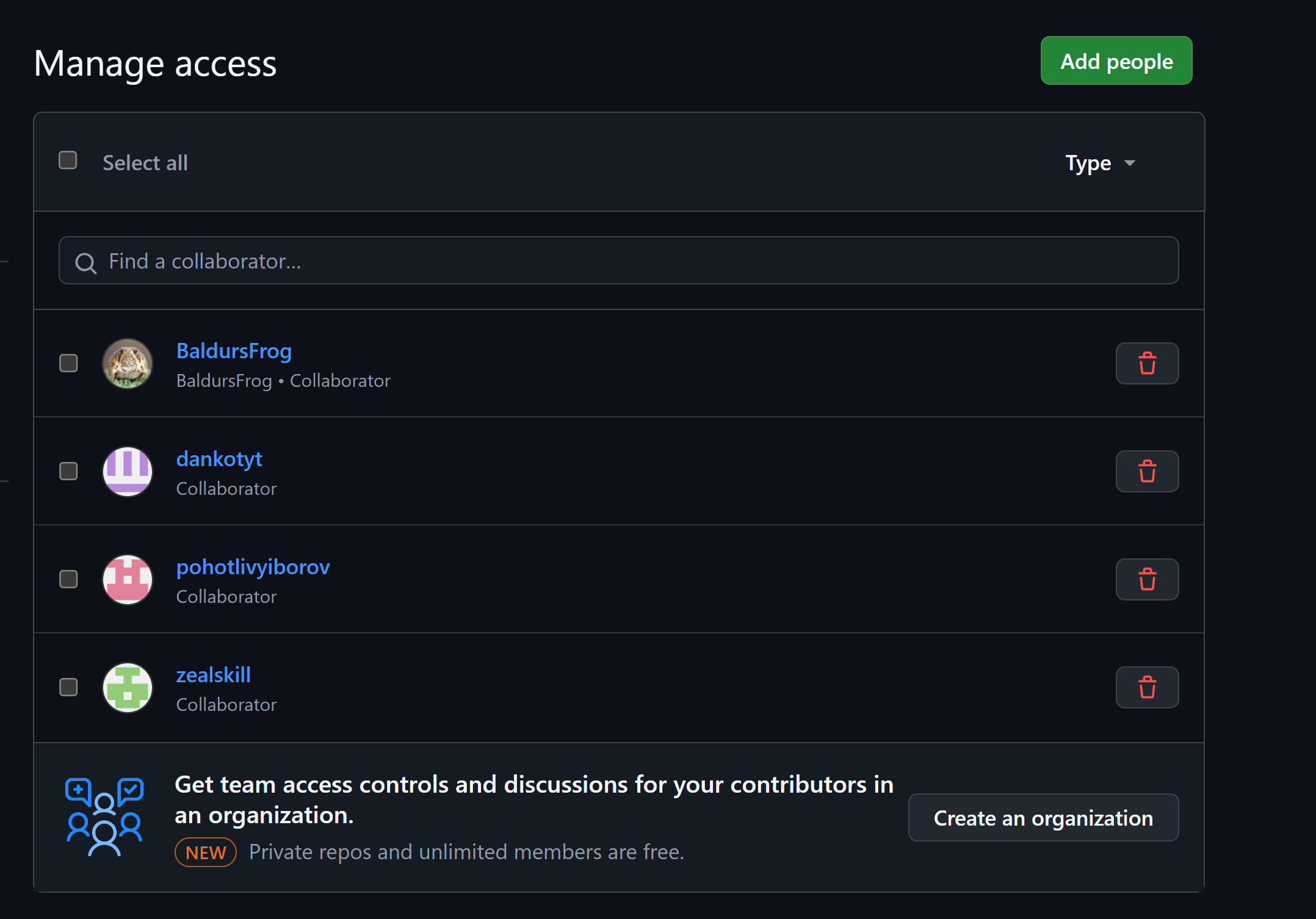
绘画14。 GitHub 上的远程测试存储库。

使用远程 - 推送，将项目上传到远程存储库：

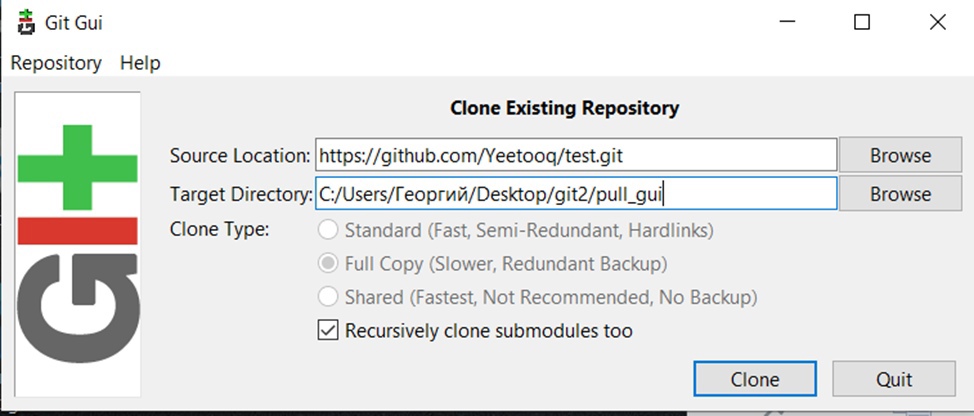


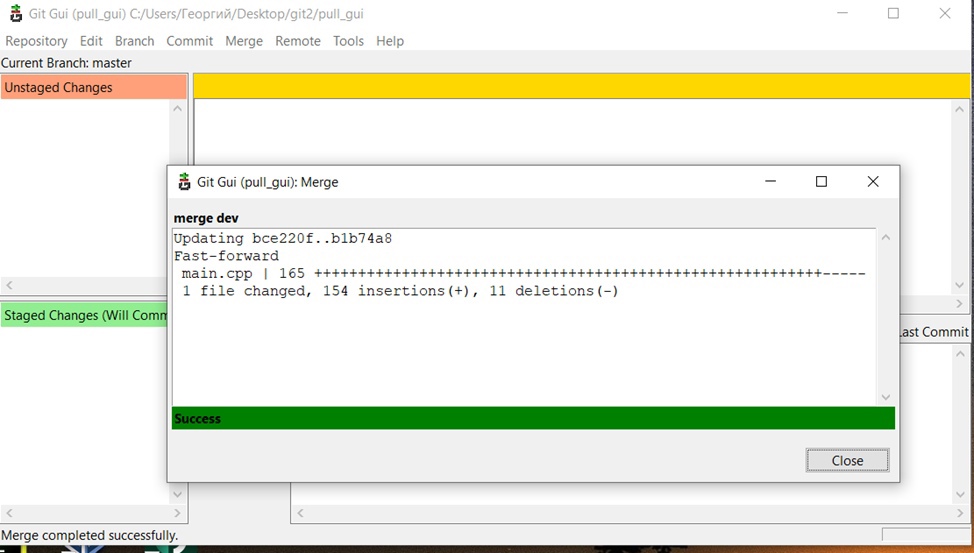
绘画15。上传到远程位置我们的项目存储库。

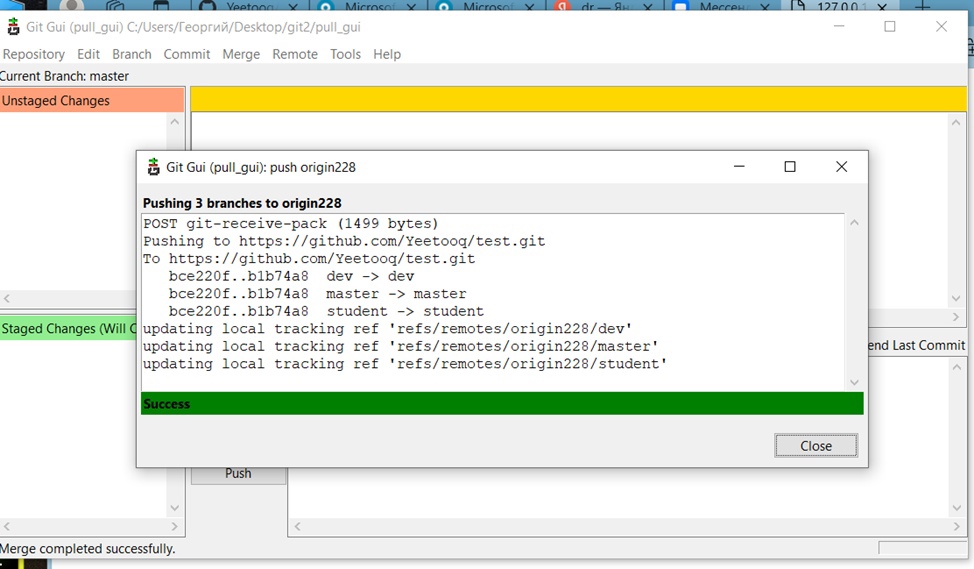
我们为整个团队提供对存储库的访问：



接下来，Georgy (BaldursFrog) 更改 main.cpp：

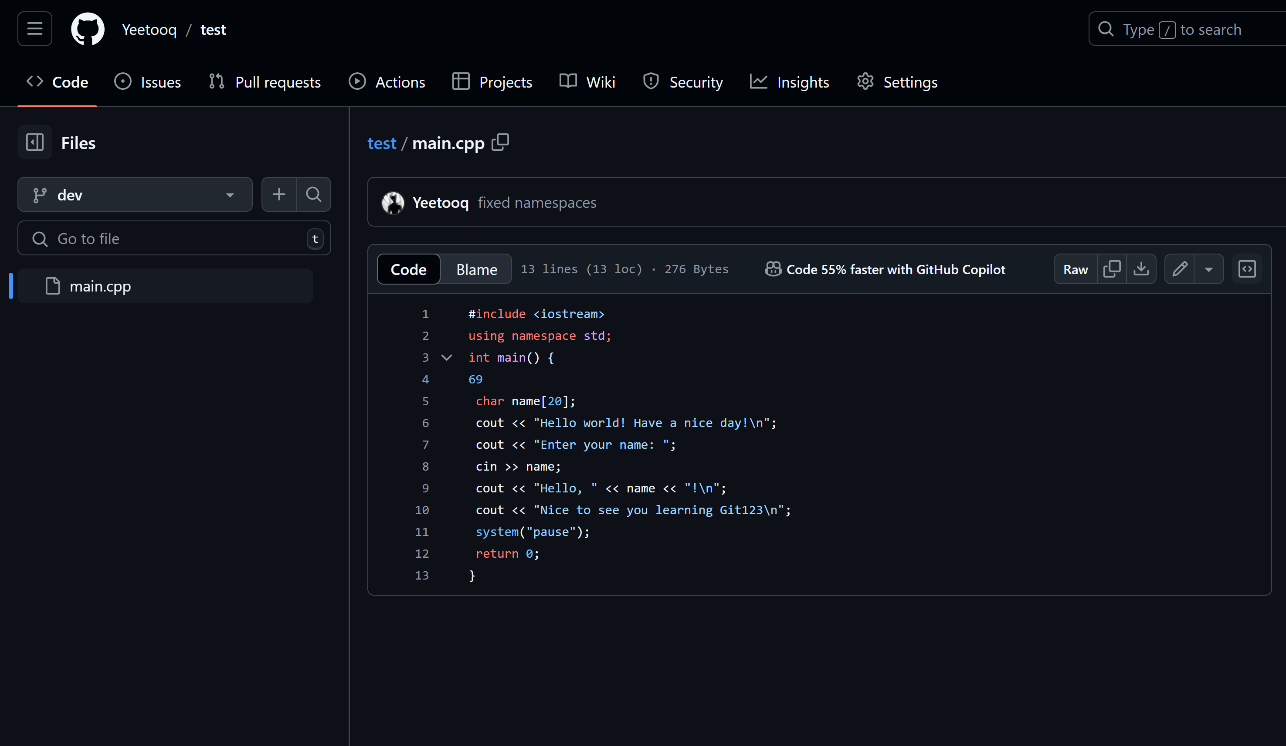






我们看到 GitHub 存储库中发生了变化。

变更前：



更改后：

