东北林业大学

毕 业 设 计

测试事务管理平台的设计与实现

学生姓名：于丽蔷

专业班级：软件工程 2017 级 4 班

指导教师：李莉 副教授

学　　院：信息与计算机工程学院

2021 年 6 月

测试事务管理平台的设计与实现

摘要

随着计算机技术的飞速发展，计算机软件复杂度不断提升，人们对软件质量有了更加严格的要求。软件测试则是软件生命周期中保证软件质量的重要过程，这个过程的核心任务在于严格执行有效的软件测试用例，及时反馈测试问题。而有效的软件测试用例设计工作的难点在于：当需求迭代速度快、需求点繁多时，通过人工构造测试用例往往出现回归无依据、基础用例遗漏、复杂场景考虑不周的情况。目前国内大多数互联网公司已经通过开发管理平台来管理测试用例，方便回归测试时进行复用，但手工录入测试用例一定程度上会加重测试工作量、延缓测试进度，因此软件测试用例管理平台利用效率不高。

本选题针对目前的软件测试用例管理平台进行进一步开发，除进行测试用例管理、方便测试用例评审和复用之外，增加两个功能：一是关联缺陷报告管理功能，其意义在于减少测试人员反馈测试问题时需描述用例场景的工作量，提高测试用例价值；二是测试用例自动生成功能，可以根据已有的科学的测试理论，实现基于软件需求的测试用例自动生成，将部分人工工作量转化为自动化工作量，提高测试工作效率，降低人为因素对测试过程的干扰。

本文针对以上需求，设计以 MySQL 作为数据库，基于 SpringBoot、Vue.js 实现 B/S 架构，并在平台上实现了基于分词算法、推荐算法和基础测试理论自动生成测试用例功能。根据实际生产需求，将系统划分为用户管理、测试用例管理、自动化测试框架管理三大模块。通过对用例图描述每个模块的基本功能，通过序列图、流程图、E-R 图进行详细设计，最终开发出测试事务管理平台，开发结束后对该系统进行了充分测试，测试结果表明本系统满足测试需求。

关键词　软件质量保障；测试用例管理；分词算法

Design and implementation of test transaction management platform

Abstract

With the rapid development of computer technology, the complexity of computer software continues to improve, people have more stringent requirements for software quality. Software testing is an important process to ensure the quality of software in the software life cycle. The core task of this process is to strictly implement effective software test cases and timely feedback test problems. The difficulty of effective software test case design lies in: when the iterative speed of requirements is fast and there are many requirements, the artificial construction of test cases often leads to the situation that there is no basis for regression, the omission of basic cases, and the thoughtlessness of complex scenarios. At present, most of the domestic Internet companies have managed the test cases through the development management platform to facilitate the reuse of regression testing. However, manual entry of test cases will increase the test workload and delay the test progress to a certain extent, so the utilization efficiency of the software test case management platform is not high.

This topic aims at the further development of the current software test case management platform, in addition to the test case management, convenient test case review and reuse, two functions are added: one is the associated defect report management function, which is to reduce the workload of describing the test case scenario when the testers feed back the test problems, and improve the value of test cases; the other is the test case self-management function Dynamic generation function can automatically generate test cases based on software requirements according to the existing scientific testing theory, which can transform part of the manual workload into automatic workload, improve the testing efficiency and reduce the interference of human factors on the testing process.

According to the above requirements, this paper designs MySQL as the database, implements B / S architecture based on Springboot and Vue.js, and based on word segmentation algorithm, recommendation algorithm and basic test theory, the function of automatic test case generation is realized on the platform. According to the actual production demand. According to the actual production requirements, the system is divided into three modules: user management, test case management and automatic test framework management. Through the use case diagram to describe the basic functions of each module, through the detailed design of sequence diagram, flow chart and E-R diagram, the test transaction management platform is finally developed. After the development, the system is fully tested, and the test results show that the system meets the test requirements.

Keywords Software quality assurance; Test case management; Word segmentation algorithm

目录

1 前言

1.1 课题研究背景及意义

随着计算机技术的飞速发展，计算机软件复杂度不断提升，人们对软件质量有了更加严格的要求。随之而来的问题就是软件从业者如何保证软件质量，保证用户信息和资产安全，提高用户产品使用体验。因此，软件测试行业应运而生。

软件测试是软件生命周期中保证软件质量的重要过程，这个过程的核心任务在于严格执行有效的软件测试用例，及时反馈测试问题。因此，通过开发可视化测试用例管理平台完成测试事务的跟进和合作十分重要。

1.2 国内外研究现状

在国外一些软件行业发达的国家，软件测试技术已经发展了非常长的时间，也相对国内得到更多重视。在一些大型软件系统开发公司，测试人员在其员工中占有相当大比重，各种测试软件、自动化测试工具应运而生。目前国外同类软件主要有 Rational 公司的 SQA Manager 产品，它是 SQA Suite 测试软件包的一部分，该软件包以测试工具 SQA Robot 和 SQA LoadTest 为主。SQA Manager 一般用做和测试工具的结合使用，为英文系统，因此产生的各类报告格式西化，没有测试案例具体步骤的管理查询，而且是以客户端的形式呈现的。在产品定位上，面向高端客户，价格昂贵，SQA Manager 作为 SQA Suite 软件包的一部分捆绑出售，不能单卖，用户购买该软件包后经常只需用其中一部分功能，造成不必要的开销[1]。

我国的软件测试技术于上世纪八十年代起源，并伴随着软件工程领域的研究发展而发展[2]。近年来，国内软件行业发展也越来越迅速，但是国内软件测试水平在国际上也还是属于较年轻类型，具体表现在对软件测试还不够重视，测试单一化，质量监督体系不够完备，自动化程度不够高等方面，软件测试人才缺口也是非常大。当然现今也有很多企业由起初的“重研发，轻测试”逐渐转变，软件测试的地位也逐渐提高，会有逐步完备的测试管理体系，不断改进的自动化测试工具，日益成熟的测试技术，软件测试将得到更多重视。本人在实习过程中，测试相关工作采用的是 i-Case 和 Bits 管理系统，上述系统主要针对测试用例的增删改查、导入导出等功能提供了解决方案，具有一定通用性，但功能单一，且手工录入测试用例工作量大，可能导致覆盖率降低；执行状况记录功能没有得到重视，不能及时更新执行结果，会导致测试的随机性和盲目性。

1.3 课题研究目标与内容

基于以上，本课题预期目标设计并实现一个基于 Web 的测试用例管理及生成系统，针对需求实现测试用例管理和缺陷问题跟进，兼容自动化测试框架管理，实现测试流程闭环。

该系统宏观分为用户管理、测试用例管理、自动化框架管理三个模块，其中用户管理基于企业应用实际，主要进行用户信息记录和权限分配；测试用例管理主要进行测试用例的生成、维护、统计和导入导出功能，除此之外，可根据实际执行情况修改执行状态和实际结果，并关联缺陷报告；自动化框架管理基于 GitHub 实现自动化测试代码的管理并统计代码覆盖率情况、生成自动化测试报告，自动化测试报告使用 JaCoCo 插件，通过插桩的方式来记录覆盖率数据，生成覆盖率报告 index.html 文件，保存至对应自动化测试用例代码目录下，形成自动化测试报告。

1.4 论文组织结构介绍

本文论文结构基于软件工程开发模型中的瀑布模型，共分为 8 章。

第 1 章：绪论，介绍本平台的研究背景、研究现状和课题主要研究目标。

第 2 章：相关理论基础与关键技术综述，介绍在本项目开发过程中使用的主要开发技术和核心算法基本原理。

第 3 章：可行性分析，主要基于作者实习期间的工作经历，介绍本问题的可行性。

第 4 章：概要设计，介绍该平台的功能结构设计方案、数据库设计方案等。

第 5 章：详细设计，介绍文中涉及到的界面、业务逻辑、核心算法和数据层的具体实现方式。

第 6 章：系统功能实现与部署，介绍界面、功能和核心算法具体实现及演示。

第 7 章：测试，介绍平台各项功能的单元测试、集成测试等，用于验证平台功能。

第 8 章：结论，对全文所做的工作进行全面总结。

1.5 本章小结

本章主要描述了课题背景，并详细讨论课题的意义以及相关课题的研究现状，还明确了主要的研究目标和内容，并介绍了论文的组织结构，对整个课题的工作进行全面概述。

2 相关理论基础与关键技术

2.1 基础测试理论

2.1.1 软件测试流程

根据作者实习经历，软件测试工作的主要流程为：需求评审、技术评审、制定测试用例、评审测试用例、冒烟测试、正式测试、回归测试、编写并提交测试报告。

在本课题中，测试用例通过关联需求文档和缺陷报告，形成测试流程闭环，提高测试效率。

2.1.2 主要测试方法

从是否关心软件内部结构和具体实现的角度划分，测试方法主要有白盒测试和黑盒测试。白盒测试方法主要有代码检查法、语句覆盖等，黑盒测试方法主要包括等价类划分法、边界值分析法、判定表驱动法等[3]。除此之外，测试方法按软件特性还可分为功能测试、性能测试、兼容性测试、安全测试等。

在本课题中，根据以上测试方法实现测试用例自动生成功能，功能测试对于取值范围使用边界值分析法、对于字符串输入采用等价类划分法，对于参数组合采用判定树/判定表法，对操作流程判断采用分支覆盖法等；性能测试提取并发数、并发持续时间、业务类型及业务占比、生产环境基础数据量、预期响应时间、系统其他特殊性能值需求（如 net I/O 不能占用带宽 1/2）等；兼容性测试可根据自动爬取网络上主流的浏览器类型、操作系统、手机端机型的统计数据和导入项目组在 APP 启动时埋点生成的数据分析报表资源，生成推荐测试列表；安全测试关注输入内容的敏感信息加密、批量操作可行性、密码的 SQL 注入等情况。

2.1.3 测试用例设计

一条完整的测试用例，一般包括用例编号、用例标题、前提条件、操作步骤、预期结果等[4]，需要满足以下原则：

1.全面性：输入数据要包括合法的、边界内的、常规的数据，也要包括非法的、边界上的、超过边界的、不合理的数据。

2.代表性：选取其中具有代表性的数据作为输入数据，减少测试用例的冗余性。

3.可判定性：每个用例必须要有明确的输出结果，用来判断实际结果是否符合预期，进而判定软件质量。

4.可操作性：需要描述清楚操作步骤，不同步骤对应不同的输出结果，在测试前做好足够的准备工作，以提高测试的效率。

5.可再现性：在执行测试用例的过程中，相同的测试用例系统所执行的测试结果是相同，以便复现和定位问题[5]。

在本课题中，基于测试用例内容和设计原则的需求，进行了相关数据库设计，确保系统的实用性。

2.2 分词算法

分词方法有标准分词、NLP 分词、索引分词、N-最短路径分词、CRF 分词以及极速词典分词等。

在本课题中，通过使用 Python 中文分词软件中的“结巴”分词，从需求文档中提取测试点关键字，根据关键字匹配测试用例，实现测试用例推荐功能[6]。

2.3 关联规则挖掘

2.3.1 关联规则挖掘基本概念

关联规则挖掘是从大量数据中挖掘出变量之间的相互联系的方法，是数据挖掘中最活跃的研究方法之一。关联规则挖掘的对象是事务数据库的记录，一个事务数据库中的关联规则挖掘可以用以下形式表示：

设 $I=\{i_1,i_2,...,i_m\}$ 是一个由 m 个不同项目组成的集合，$D=\{t_1,t_2,...,t_n\}$ 是一个由一系列具有唯一标识符的 TID 事务组成的事务数据库，每一个事务 $ti(i=1,2,...,n)$ 都对应 I 上的一个子集[7]。

$k-$项集：$k-$项集是一个包含数据项的集合。

支持度：事务集 D 中的项集 X 的支持度记为 $\sup(X)$，表示包含 X 的事务在事务集 D 中所占的百分比，见公式 2.1。

$$\sup(X)=\frac{\left\|\{t_i \mid X\subseteq t_i,t_i\in D\}\right\|}{\|D\|} \tag{2.1}$$

其中 $X\subseteq I$，$\|\cdot\|$ 表示包含项集 X 的事务数量。

若有两个项目集 X 和 Y，$X\cap Y=\varnothing$，交集为 $X\cup Y$，则关联规则 $X\Rightarrow Y$ 的支持度公式如公式 2.2 所示。

$$\sup(X\Rightarrow Y)=\frac{\left\|\{t_i|(X\cup Y)\subseteq t_i,t_i\in D\}\right\|}{\|D\|} \tag{2.2}$$

置信度：置信度表示包含 X 和 Y 的事务数与包含 X 的事务数值比，见公式 2.3。置信度体现了关联规则的可靠性，置信度越高说明 Y 在包含 X 的事务中出现的可靠性越大。

$$conf(X\Rightarrow Y)=\frac{\left\|\{t_i|(X\cup Y)\subseteq t_i,t_i\in D\}\right\|}{\left\|\{t_i|X\subseteq t_i,t_i\in D\}\right\|} \tag{2.3}$$

频繁项集和最大频繁项集：对于事务数据库 D 中所有满足最小值尺度的项目集称为频繁项目集，否则称为非频繁项目集。若频繁项集 Ii 的所有超集都是非频繁项集，则称 Ii 为最大频繁项集。

2.3.2 关联规则挖掘算法

关联规则一般用最小支持度和最小置信度来进行筛选，满足一定阈值的事物会被筛选出来。进行关联规则挖掘的首要步骤是进行频繁项挖掘，经典的频繁项挖掘算法有 Apriori 算法[8]，该算法自底向上逐层搜索迭代，利用 $(k-1)-$项集去搜索 $k-$项集，直到所有项数的频繁项都被挖掘出来。本文将使用这种算法对测试用例进行推荐。

2.4 SpringBoot 框架

Spring Boot 是用来简化新 Spring 应用的初始搭建以及开发过程，该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置[9]。

在本课题中，编码通过使用 Spring Boot 框架，减少配置配置 web.xml、配置数据库连接、配置 Spring 事务、配置加载配置文件的读取、配置日志文件等工作，提高了开发效率。

2.5 Vue.js 框架

Vue.js 是一个提供了 MVVM 风格的双向数据绑定的 Javascript 库，专注 View 层，它被设计为可以自底向上逐层应用，便于与第三方库或既有项目整合[10]。

本课题在 MVVM 架构下，ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，View 和 Model 之间的同步工作完全是自动的，无需人为干涉。因此开发者只需关注业务逻辑，不需要手动操作 DOM，不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

2.6 MySQL

MySQL 是一个关系型数据库管理系统，它是目前最流行的关系型数据库之一，在 WEB 应用方面，MySQL 是最好的关系数据库管理系统)应用软件之一[11]。

在本课题中，MySQL 搭建关系型数据库，主要用于储存用户信息和测试用例信息，并为平台提供数据源，以保证项目的正常运行。

2.7 本章小结

本章重点介绍了在本课题中依赖的相关理论基础，使用工具和相关技术，并阐述了相关算法的基本原理，后文则会基于本章的内容进行引申与应用。

3 需求分析

本课题所需要构造的测试事务管理平台，主要用户是开发人员和测试人员。因此，用户的整体需求基本决定了平台的功能。本文基于对作者实习期间的工作经历，完成了该平台的可行性研究与需求分析。

3.1 平台分析概述

对本课题测试事务管理平台进行分析，系统功能主要划分为三大部分：用户管理、测试用例管理、自动化框架管理。上述三个部分之间相互独立、相互联系。

从总体框架设计上看，系统采用面向对象的编程范式，本着低耦合、高内聚的设计原则[12]，采用可视化的界面设计，以达到预期效果，预留足够的系统接口，保证了代码的可复用性和系统的可拓展性，为后期业务迭代提供可能。

3.2 平台可行性分析

可行性分析是软件生命周期中的最初阶段。软件开发设计人员一般会在软件需求定义之前，对软件开发的可行性进行全方位考虑，以便于直接驱动后期各项开发与维护工作的开展。本文重点围绕技术可行性、经济情况可行性与社会环境可行性等方面进行可行性的论证与分析。

3.2.1 技术可行性

在本课题中，技术可行性是指功能的实现和质量的保证。

对于本课题来说，使用者主要是软件测试人员和开发人员，用户使用过程中不存在高性能需求，数据存储方面不需要采用大型数据库。对于初步 B/S 框架而言，使用 SpringBoot 和 Vue.js 进行实现，核心功能通过“结巴”分词软件和推荐算法实现即可，不存在功能实现方面的困难。除此之外，作者实习过程中已积累近一年的测试经验，对于质量保障方面具有一定的理解，可在测试过程中对软件质量进行把关。

综上，可在原定开发时间中完成课题功能需求的实现，并保证软件的质量，本平台在技术实现层面具备较强的可行性。

3.2.2 经济可行性

在本课题中，经济可行性主要是包括成本和收益。

对于软件成本而言，本系统 B/S 架构主要基于 IntelliJ IDEA 进行开发，使用教育邮箱可避开软件收费问题，其他软件则均为开源版本或者免费版本，不存在经济额外支出。对于硬件成本而言，基于作者已有笔记本电脑，软件部署在本地 Windows10 系统。

综上，该课题的实现过程中不会产生经济开销，本平台在经济上具有较好的可行性。

3.2.3 社会因素可行性

在本课题中，社会因素可行性主要是包括法律可行性、使用可行性。

对于法律因素，所有开发软件都选用正版，开发过程和开发内容严格遵守有关知识产权法和软件行业的相关法律法规，不会侵犯任何个人、集体、国家的利益。使用者主要为团队内部人员，具备专业技术素养，简单介绍即可使用。

综上，该课题开发合法合规，且具有易用性，本平台在社会因素上具有较好的可行性。

3.3 平台需求分析

本平台主要用户为开发人员和测试人员，因此，下面对平台的主要用户和主要功能进行需求分析。

3.3.1 功能需求

基于作者实习经历中的使用需要，得出平台的需求如下：

（1）用户管理模块：本平台主要包括三类用户，管理员、开发人员与测试人员。管理员登录后，可以进行用户账号和权限分配；测试人员登录后，可以修改个人信息，进行测试用例和自动化框架维护；开发人员登录后，可以修改个人信息，进行缺陷报告维护和测试用例查看。

（2）测试用例管理模块：该模块包括测试用例的新建（包括手动创建和自动创建）、批量导入测试用例、测试用例的修改、导出测试用例、删除测试用例、测试用例查询、测试用例执行状态标记、测试用例关联缺陷报告等子功能。

（3）缺陷报告管理模块：该模块主要实现缺陷报告的新增、修改、删除、查询功能，方便开发人员和测试人员对缺陷问题进行跟进。

（4）自动化测试框架管理模块：该模块主要实现自动化测试框架的代码管理，并通过 JaCoCo 插件惊醒覆盖率统计和测试报告的生成。

3.3.2 非功能需求

对于非功能需求，主要指的是除了功能外的其他软件需求，一般包括性能需求、硬件需求等内容，在本课题中，主要包括以下需求：

（1）性能需求

针对于搜索，应当保证较短的响应时间，尽最大努力减少数据库性能的损耗。系统应当能够承受瞬时请求 30 次/秒的压力，且平均在 2 秒内(外部联络接口在 10 秒内)返回结果。

（2）硬件需求

系统应当具备在主流计算机进行使用的能力，应当考虑系统整体的兼容性。

（3）其余需求

系统本身应当具备较强的容错性，保证系统本身的稳定运行。系统同时应该采取相关技术来保证数据的安全性，以及系统平台的可靠性。

3.4 平台建模

根据上述分析，本课题建模主要采用交互模型中的用例建模和行为模型中的数据流建模。

3.4.1 用例建模

用例建模主要包括正在开发的系统与其他系统之间的交互。有助于识别用户需求，解决系统间可能产生的交流问题。

在本课题中，平台主要涉及到的用户身份包括管理员、测试人员和开发人员。预计不同角色设计的功能共分为三大类：用户管理功能、测试用例管理功能、缺陷报告管理功能。用户管理模块主要是管理员使用的基础服务，用于管理团队人员信息和统计数据等；测试用例管理模块包括文本用例和自动化测试用例框架管理，主要包括测试用例的生成、维护等基础功能和代码的迭代；缺陷报告管理是基于测试用例管理中的关联缺陷报告功能，提供给开发人员进行缺陷处理的入口。详细的用户用例图如图 3-1 所示。

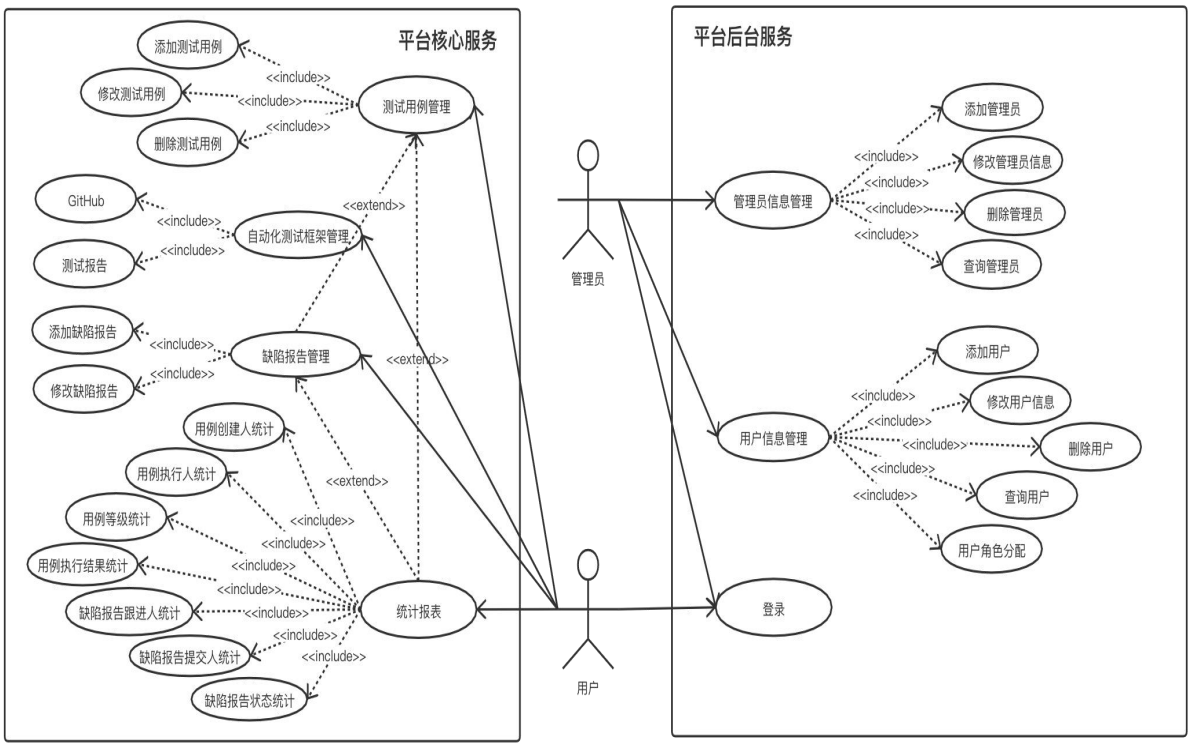


图 3-1 平台用户用例图

3.4.2 数据驱动建模

数据流建模一般基于数据流图（Data Flow Diagram，英文简称 DFD）以图形方式描绘数据的流动及处理过程，反映了系统必须完成的逻辑功能[13]。

在本课题中，主要流程为需求评审和技术评审后，测试人员制定测试用例，进行测试用例评审，开发人员冒烟测试通过后，测试人员进行正式测试，对于缺陷问题编写并提交缺陷报告，待开发人员进行处理后，测试人员进行回归测试。整个平台中主要的数据流走向大致可以分为以下三个数据流图。

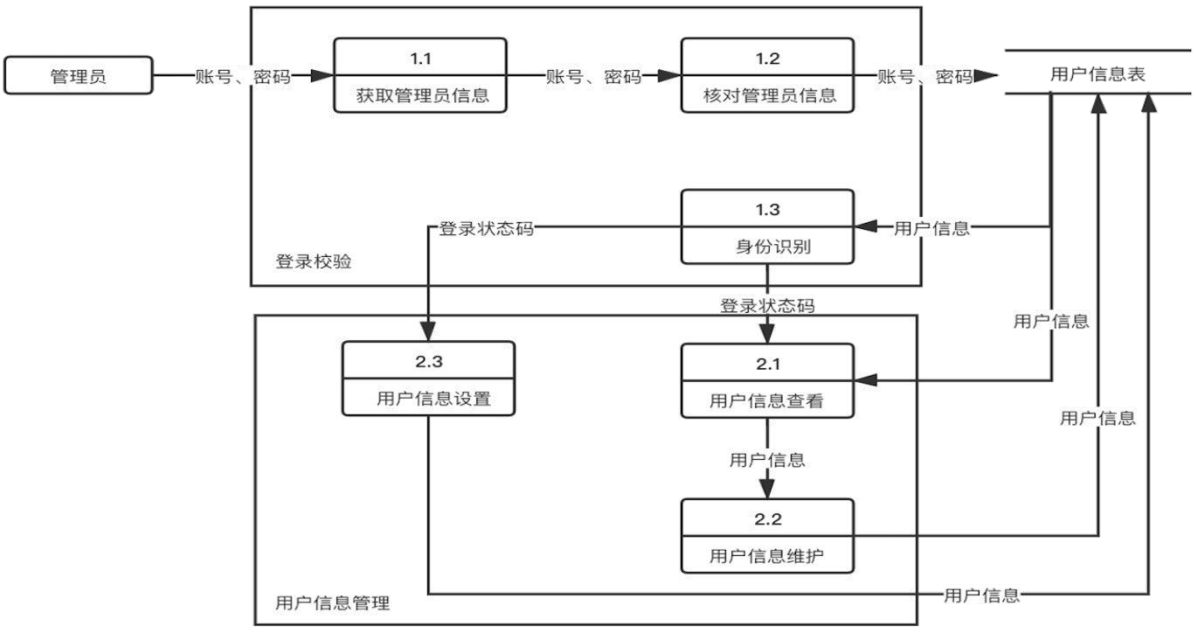


图 3-2 管理员数据流图

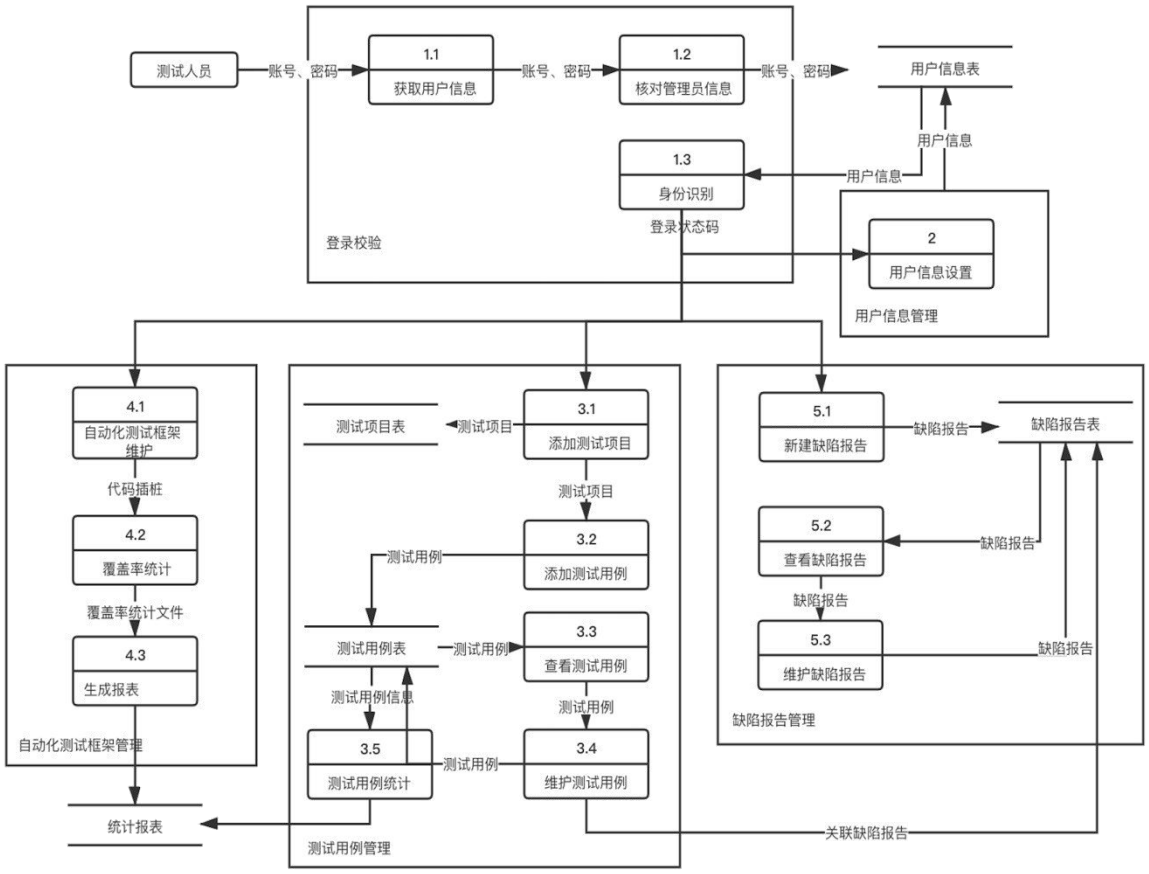


图 3-2　测试人员数据流图

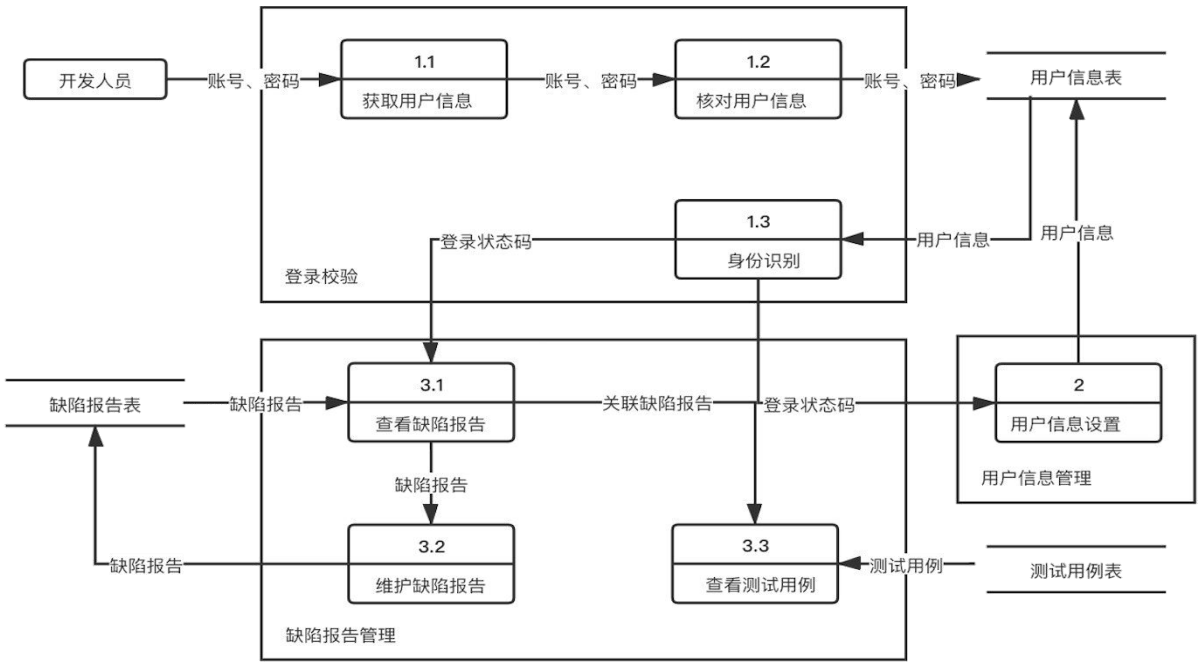


图 3-3　开发人员数据流图

3.5 本章小结

本章对平台搭建的可行性进行了分析，同时进行了基于作者实习经历进行了需求分析，并完成了用例建模和数据流建模。下文将基于上述分析进一步阐述平台的构建。

4 概要设计

概要设计，主要是对需求分析之后进行的下一步概要性的工作。设计几乎完全决定了整个软件构造的成或败，一个良好的设计能够有效提高软件的易用性以及可维护性。

在本章中，我们主要对系统的系统结构、功能需求和数据库进行设计，以便保证该系统在宏观上具备一定优秀的特性。

4.1 平台开发结构设计

系统开发结构设计的具体流程如图 4-1 所示。

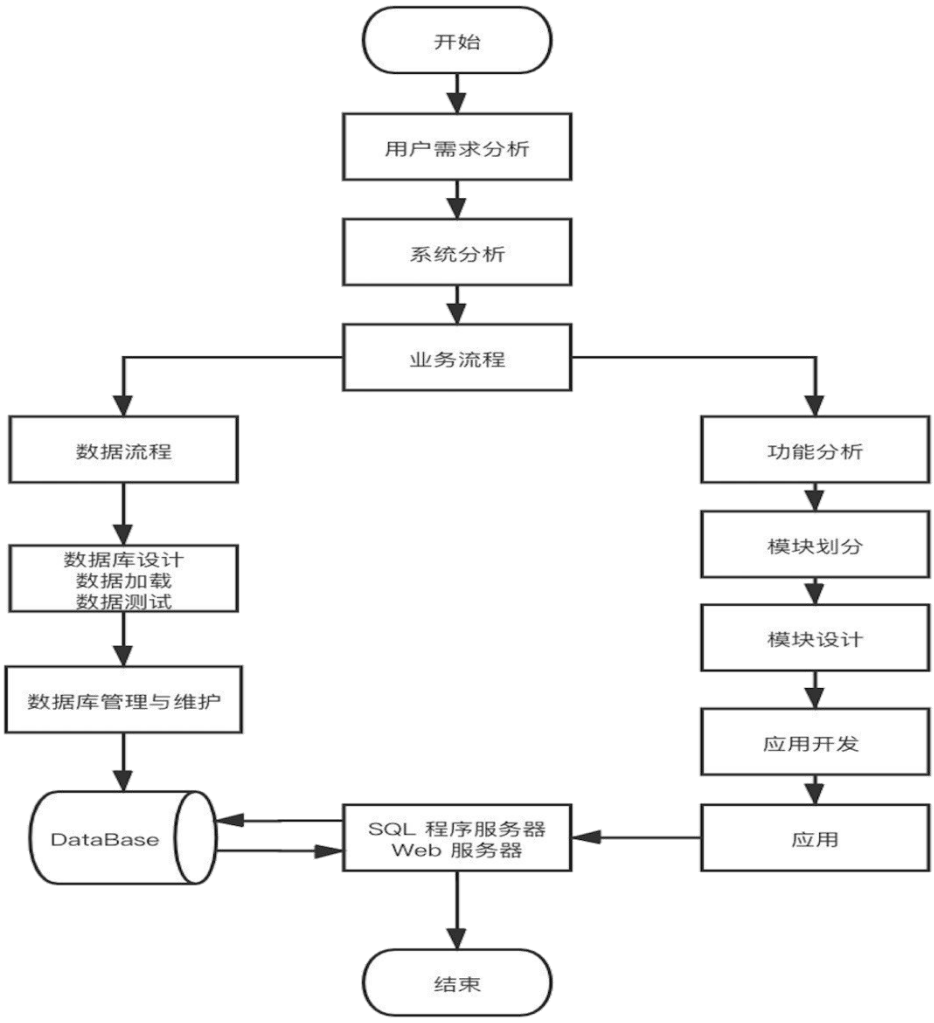


图 4-1 系统开发结构设计图

4.2 平台总体框架设计

本系统采用三层架构模式，将系统划分为表现层、业务逻辑层以及数据访问层，如图 4-2 所示。在表现层中使用 Vue.js 框架，对交互式可视化界面进行描述，实现永华界面和后台代码的分离，最终返回给用户需要的信息。Web Service 层在本系统中作为业务逻辑层出现，在 Web Service 中定义需要的逻辑方法，用户通过表现层将命令发送给逻辑层，调用相关的逻辑方法来更新数据层定义的数据，将操作结果通过表现层返回给用户。数据访问层的作用即是与数据库进行交互，生成对应的数据实体类，完成对数据库的操作。

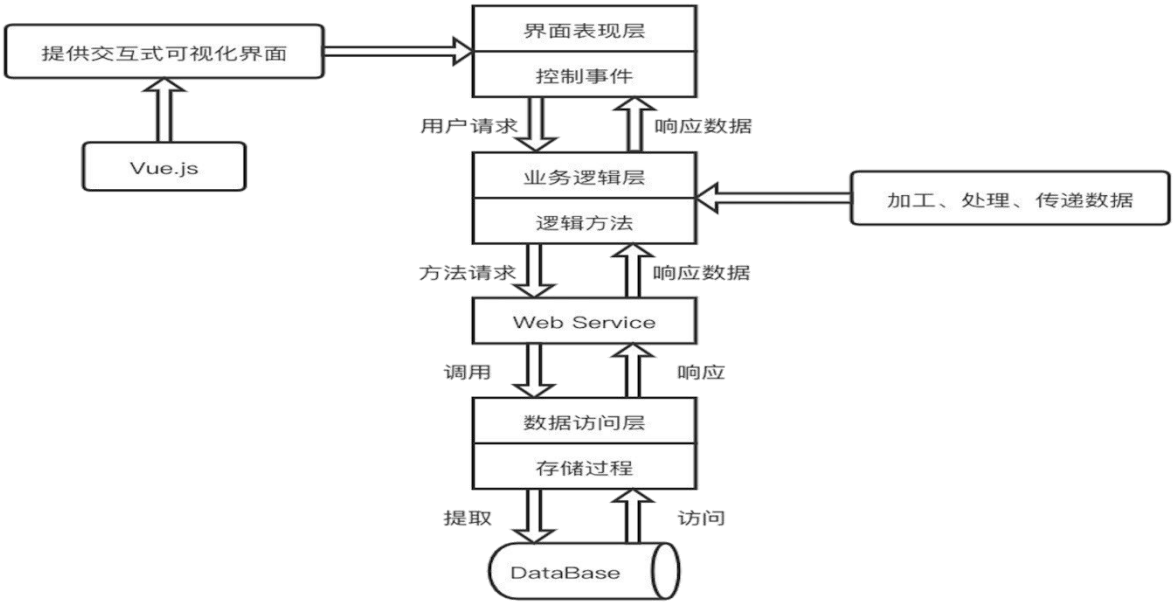


图 4-2 系统三层架构图

4.3 平台功能模块设计

功能结构设计主要用于对软件的功能点和从属关系进行全面的设计。在进行设计时，通常会采用功能结构图进行合理描述。功能结构图采用最简单的树形表现方式来对软件的功能进行全面梳理。在本课题中，主要的功能结构图如图 4-1 所示。

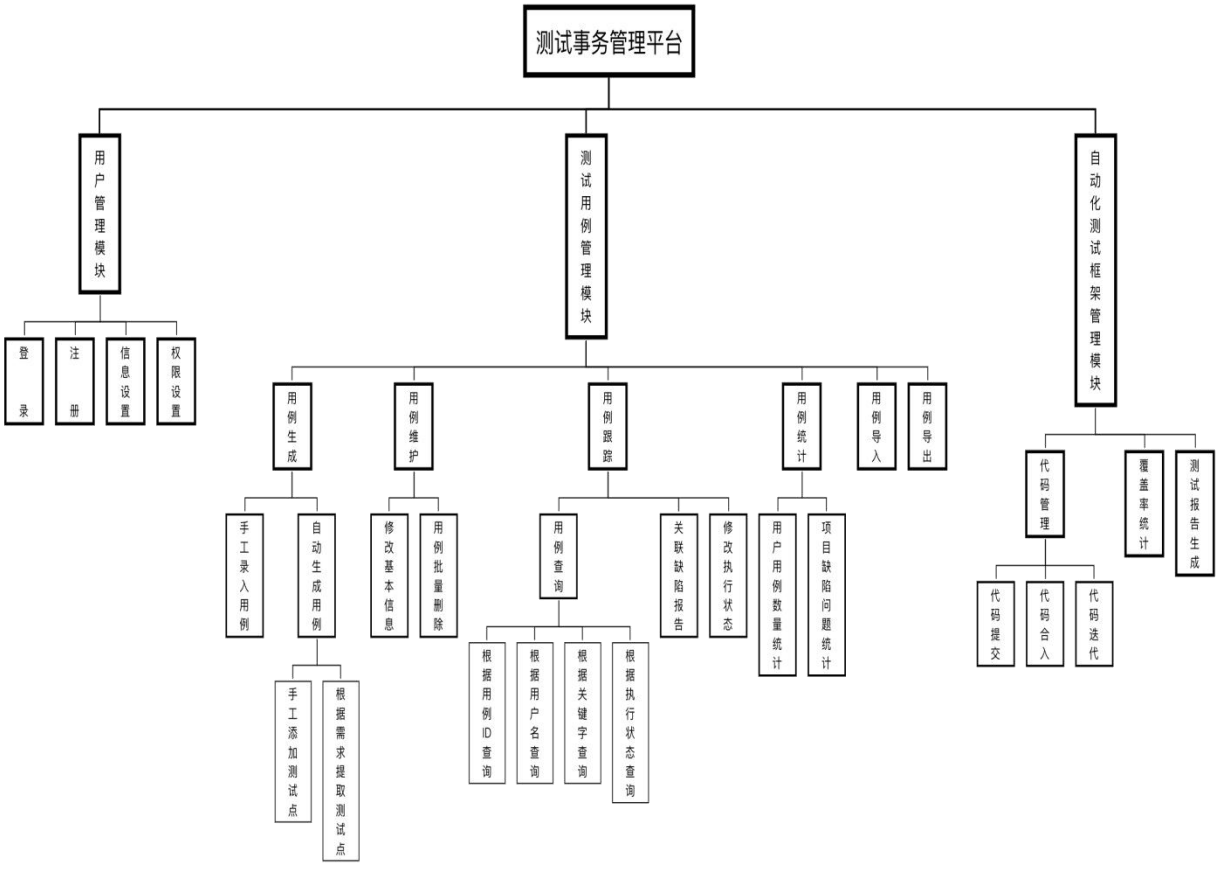


图 4-3 系统核心功能结构图

该课题所搭建的核心平台——测试事务管理平台包括三个主要功能模块：用户管理模块、测试用例管理模块和自动化测试框架管理模块，除此之外，根据实际需求单独进行缺陷报告管理。

4.3.1 用户管理模块

本平台用户管理模块主要包括登录、用户信息设置，管理员还可进行用户账号分配和权限设置，方便开发人员和测试人员可以根据不同的权限跟进不同项目。

4.3.2 测试用例管理模块

测试用例管理模块的主要功能点如下：

（1）用例显示：用户打开测试用例模块，目录中显示为所有项目列表，项目下为测试用例，根目录可以显示所有的测试用例。

（2）用例生成：

（a）手工录入测试用例：当新建测试用例时，会按规则自动生成一个唯一的 ID，便于查询和唯一标识，然后依次填入用例标题、用例描述、测试步骤、预期结果、相关依赖（脚本、数据等）、测试分类等信息，并确认新建，信息完整性校验之后，则新建成功。其中用例信息自动填写创建人和执行人字段，创建人为录入基础信息的用户，执行人为修改执行状态的用户。

（b）自动生成测试用例：根据需求文档分词匹配已经有的测试用例集推荐生成，或根据测试理论实现自动生成，例如：功能测试对于取值范围使用边界值分析法、对于字符串输入采用等价类划分法，对于参数组合采用判定树/判定表法，对操作流程判断采用分支覆盖法等；性能测试提取并发数、并发持续时间、业务类型及业务占比、生产环境基础数据量、预期响应时间、系统其他特殊性能值需求（如 net I/O 不能占用带宽 1/2）等；兼容性测试可根据自动爬取网络上主流的浏览器类型、操作系统、手机端机型的统计数据和导入项目组在 APP 启动时埋点生成的数据分析报表资源，生成推荐测试列表；安全测试关注输入内容的敏感信息加密、批量操作的可行性、密码的 SQL 注入等情况。

（3）用例维护：

（a）修改基本信息：可以进行上述测试用例基本信息的修改；

（b）修改执行状态：测试完成后对应编辑执行状态，方便查看是否测试通过；

（c）关联缺陷报告：如测试不通过，测试人员可提交相应缺陷报告，记录对应链接在测试用例中，方便后入回归；

（d）批量删除用例：如录入有误，可以进行测试用例的删除，测试用例列表采用复选框，可以进行单选或批量删除。

（4）用例跟踪：采用字符串匹配算法，根据查询条件进行查询，查询条件包括：根据用例标题查询、根据用户名查询、根据用例状态查询、根据重要程度、根据用例类型查询。

（5）用例统计：

（a）用户用例数量统计：采用 SQL 语句，根据数据库中测试用例表下的创建人字段，统计用户生成的测试用例数量；

（b）项目缺陷报告统计：根据缺陷报告字段，统计改项目目录下的缺陷报告数量。

（6）用例导入和导出：

可导出标准模板 Excel 表，填写完成各字段后上传至系统；也可批量选中测试用例后，导出测试用例 Excel 表。

4.3.3 自动化测试框架管理模块

自动化测试框架管理模块的主要功能点如下：

（1）代码管理：实现 GitHub 功能，GitHub 是一个面向开源及私有软件项目的托管平台，只支持 Git 作为唯一的版本库格式进行托管。测试人员通过 Git 将本地代码提交至同一个项目仓库，进行合入、更新等代码维护工作，实现自动化测试工作合作。

（2）覆盖率统计：通过 JaCoCo 插件，使用插桩的方式记录覆盖率数据。方便收集测试过程中代码覆盖情况，能够很直观展现哪些代码已经测试过，哪些没有被测试，帮助测试人员有针对性地增加用例，提升测试质量。

（3）测试报告生成：实现自动化测试用例（如单元测试）的执行，并统计执行结果形成测试报告。

4.3.4 缺陷报告管理模块

缺陷报告主要字段包括关联测试用例、问题描述、相关信息（如日志、抓包文件等）、缺陷状态、提交人和跟进人等。缺陷报告管理模块的主要功能点如下：

（1）缺陷报告生成：测试人员根据测试用例执行结果新建缺陷报告，并将缺陷报告链接关联至测试用例，方便测试人员后续回归测试，并减少了描述复现场景的工作量，开发人员则可以通过查看缺陷报告关联的测试用例进行问题复现、排查。

（2）缺陷报告维护： 除了修改缺陷报告的基本信息外，主要针对缺陷问题的状态进行修改，开发人员进行修复后则可以修改缺陷状态为待验证，测试人员回归测试后即可将缺陷状态修改为已修复。

4.4 平台数据库设计

针对于平台的储存需求，本设计需要引入关系型数据库来对数据进行储存。在多种关系数据库中，MySQL 由于其优秀的性能、便捷的安装和使用方法、以及免费的特性被众多开发者选择。在本课题中，我们选择这一数据库来对数据持久层中关系型数据库进行实现，并对数据库进行建模。

4.4.1 E-R 设计

实体联系图(Entity-Relationship Model，简称 E-R 图)主要用来描述实体及其之间的关系，其是数据库概念设计中较为常用的图形设计方法之一。在本设计中，基于 E-R 图，表示各个实体之间的关系，如图 4-6 所示：

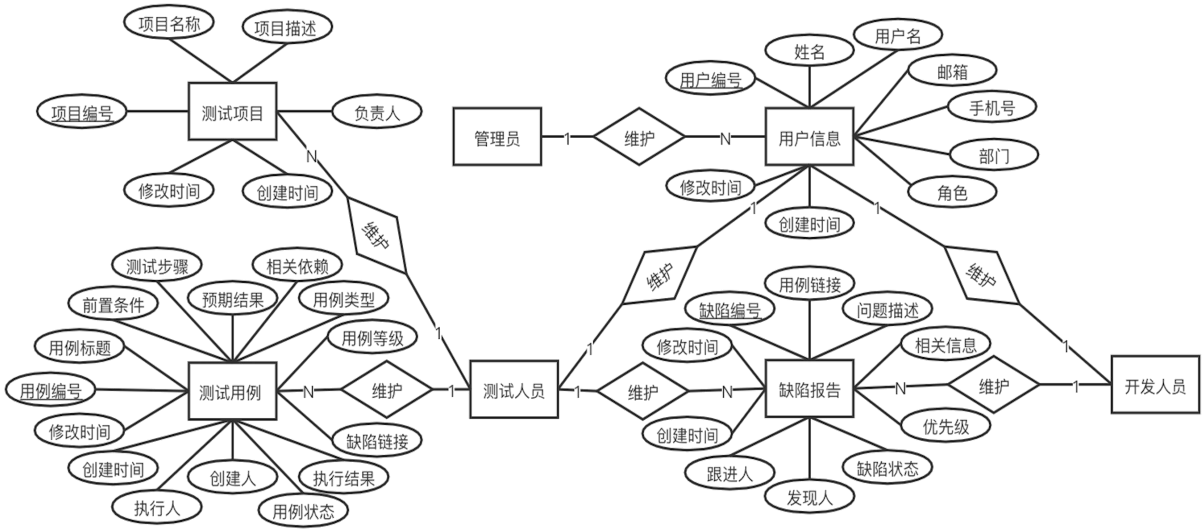


图 4-4 平台实体-联系示意图（E-R 图）

在本平台中，主要有管理员、测试人员、开发人员、用户信息、测试项目、测试用例、缺陷报告等实体，各个实体均有其相关属性和联系。接下来本文将详细阐述关系型数据库的存储结构。

4.4.2 数据库表设计

数据库表设计即为平台的存储结构设计[14]。本系统的所有数据表均来源于上述实体及其多对多所产生的数据表。具体表格的设计如表 4-1 至 4-4 所示。

表 4-1 用户信息表表

代码	名称	数据类型	是否主键	注释
id	用户编号	int(11)	true	主键，自动增长
name	姓名	varchar(255)	false	非空
user	用户名	varchar(255)	false	非空，唯一索引
email	邮箱	varchar(255)	false	非空，唯一索引
telephone	手机号	varchar(255)	false	非空，唯一索引
department	部门	varchar(255)	false	非空
character	角色	int(11)	false	0 代表管理员 1 代表测试人员，默认值 2 代表开发人员
create_time	创建时间	datetime	false	
update_time	修改时间	timestamp	false	

表 4-2 测试项目表

代码	名称	数据类型	是否主键	注释
id	项目编号	int(11)	true	主键，自动增长
name	项目名称	varchar(255)	false	非空
description	项目描述	varchar(3000)	false	非空
owner	负责人	varchar(255)	false	非空
create_time	创建时间	datetime	false	
update_time	修改时间	timestamp	false	

表 4-3 缺陷报告表

代码	名称	数据类型	是否主键	注释
id	缺陷编号	int(11)	true	主键，自动增长
case	用例链接	varchar(3000)	false	相关测试用例链接
description	缺陷描述	varchar(3000)	false	非空
message	相关信息	varchar(3000)	false	允许为空
level	优先级	int(11)	false	0 代表 P0，默认值 1 代表 P1 2 代表 P2 3 代表 P3
status	缺陷状态	int(11)	false	0 代表新提交，默认值 1 代表已修复 2 代表已发布 3 代表重新打开
founder	发现人	varchar(255)	false	非空
operator	负责人	varchar(255)	false	允许为空
insert_time	创建时间	datetime	false	
updatetime	修改时间	timestamp	false	

表 4-4 测试用例表

代码	名称	数据类型	是否主键	注释
id	用例编号	int(11)	true	主键，自动增长
title	用例标题	varchar(3000)	false	非空
condition	前置条件	varchar(3000)	false	允许为空
step	测试步骤	varchar(3000)	false	允许为空
expect	预期结果	varchar(3000)	false	允许为空
dependence	相关依赖	varchar(3000)	false	允许为空
type	用例类型	int(11)	false	0 代表功能用例，默认值 1 代表接口用例 2 代表性能用例 3 代表兼容性用例 4 代表 UI 用例
level	重要程度	int(11)	false	0 代表 P0，默认值 1 代表 P1 2 代表 P2 3 代表 P3
bug	缺陷链接	varchar(3000)	false	相关缺陷报告链接
result	执行结果	int(11)	false	0 代表未执行，默认值 1 代表通过 2 代表不通过 3 代表受阻 4 代表跳过
status	用例状态	int(11)	false	0 代表草稿，默认值 1 代表通过 2 代表未通过 3 代表废弃
founder	创建人	varchar(255)	false	非空
operator	执行人	varchar(255)	false	允许为空
create_time	创建时间	datetime	false	
update_time	修改时间	timestamp	false	

4.5 本章小结

本章基于此前的需求相关分析，对系统相关功能结构、三层架构与数据库设计进行了相当详细的描述。设计结果将直接驱动详细设计和系统开发过程。与其相关的详细业务将在后文进行实现。

5 详细设计与实现

总体设计完成之后，便需要对上述提出的各项设计进行详细设计并实现。在该模块中，主要对软件的界面层、业务逻辑层和核心算法层进行详细设计与实现。

5.1 平台界面设计与实现

平台的界面设计美观、易用，直接影响到用户工作效率，下面将对平台主要模块界面进行设计与实现，平台实现示例图片将根据展示需要，进行字段展示顺序的修改。

5.1.1 用户登录功能界面设计与实现

用户登录界面需要用户输入自己的用户名、密码和验证码、角色进行登录，具体实现界面如图 5-1 所示。

图 5-1 用户登录界面

5.1.2 用户管理界面设计与实现

用户管理界面主要实现用户信息的显示，包括姓名、用户名、手机号、邮箱、部门、角色等，如图 5-2 所示。具体功能如下：

（1）添加、查找按钮显示在用户列表上方；

（2）其他功能显示在操作栏中，可以修改用户信息、设置用户密码；

（3）根据实际情况，删除用户采用禁用用户代替。

图 5-2 用户管理界面

5.1.3 测试项目菜单设计与实现

测试项目在菜单栏中作为文件夹显示，每个文件夹下用子文件夹代表该测试项目的不同模块，如图 5-3 所示，具体业务规则如下：

（1）鼠标点击根文件，显示所有测试用例；

（2）鼠标点击子文件夹，显示子文件夹中测试用例；

（3）测试项目菜单可以收起至左侧；

（4）鼠标悬浮至文件夹上显示操作按钮，包括新建、编辑和删除。

图 5-3 测试项目菜单界面

5.1.4 测试用例管理界面设计与实现

测试用例管理界面主要展示测试用例编号、状态、执行结果、标题、用例类型、重要程度、状态、执行人、更新时间、创建时间等信息，其他信息展示在详情处，如图 5-4 所示，具体业务规则如下：

（1）添加、查找、导入、导出按钮显示在用例列表上方；

（2）可在页面直接修改状态、执行结果、标题、用例类型；

（3）鼠标点击标题，可查看用例详情；

（4）悬停至标题，显示编辑、复制链接、关联缺陷报告按钮，如图 5-5 所示；

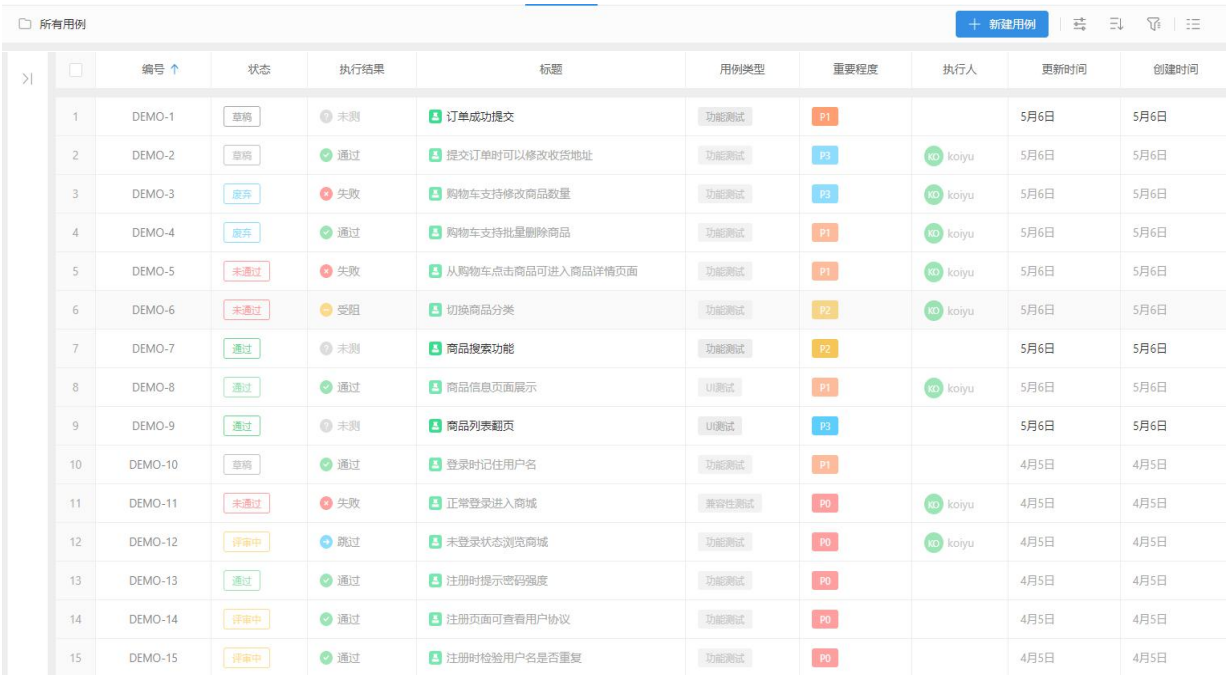


图 5-4 测试用例管理界面

图 5-5 测试用例操作界面

5.1.5 缺陷报告管理界面设计与实现

缺陷报告管理界面与测试用例页面相同，大体如图 5-6 所示，在这里不多赘述。

图 5-6 缺陷报告管理界面

5.1.6 统计报表界面设计与实现

统计报表界面如图 5-7 和 5-8 所示，主要实现了简洁直观的用例统计功能。

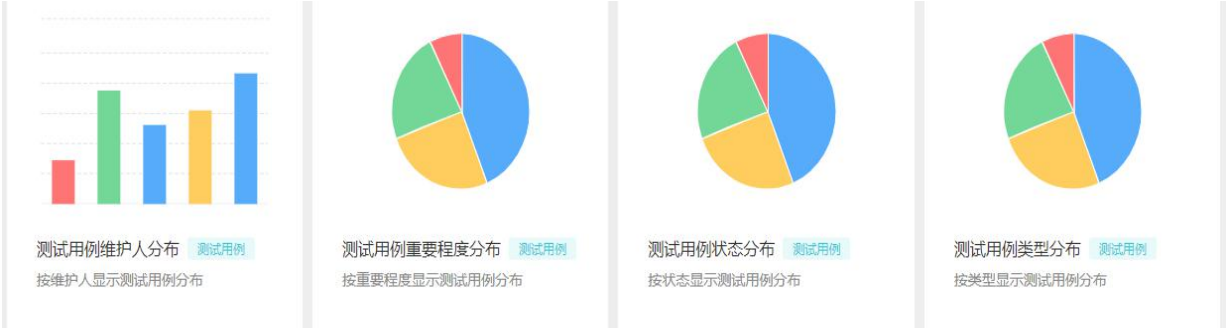
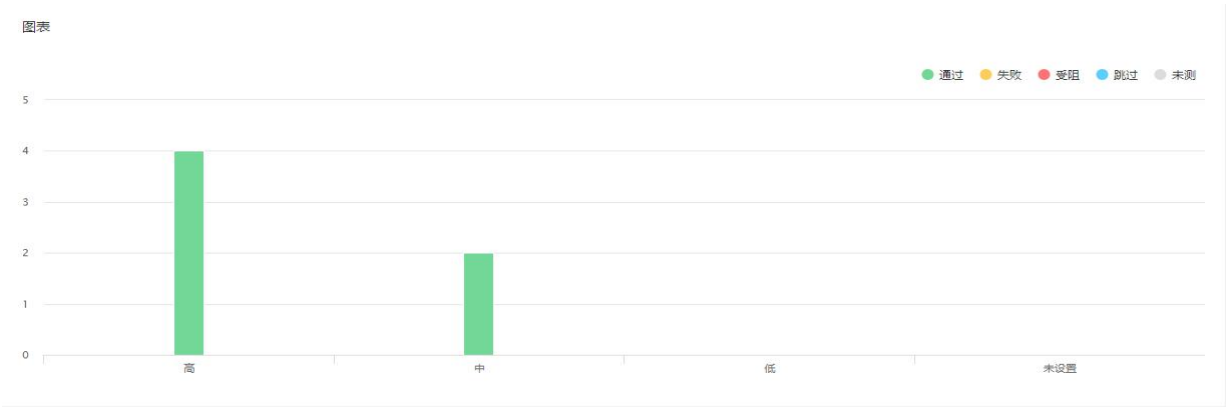


图 5-7 统计报表概览图



表格

优先级	通过	失败	受阻	跳过	未测	总计
高	4	0	0	0	0	4
中	2	0	0	0	0	2
低	0	0	0	0	0	0
未设置	0	0	0	0	0	0
总计	6	0	0	0	0	6

图 5-8 统计报表详细图出图

5.2 平台业务逻辑层设计与实现

本系统中除用户登录外存在较多的业务逻辑，包括添加、修改、查询、删除、导入、导出等。受限于篇幅限制，本文即主要通过用户登录、测试用例管理模块、自动化测试框架管理模块以及缺陷问题处理的业务逻辑展开叙述。

5.2.1 用户登录

用户登录时序图如图 5-9 所示：以下简单介绍该系统登录的判断原则：

（1）用户名及对应角色不存在，则不允许登录，提示“用户不存在！”

（2）用户名及对应角色存在，但密码错误，则不允许登录，提示“用户名与密码不匹配！”

（2）用户名及对应角色存且密码正确，但验证码错误，则不允许登录，提示“验证码错误，请重新输入！”

（4）账号、角色、密码和验证码正确，则跳转到用户角色对应界面。

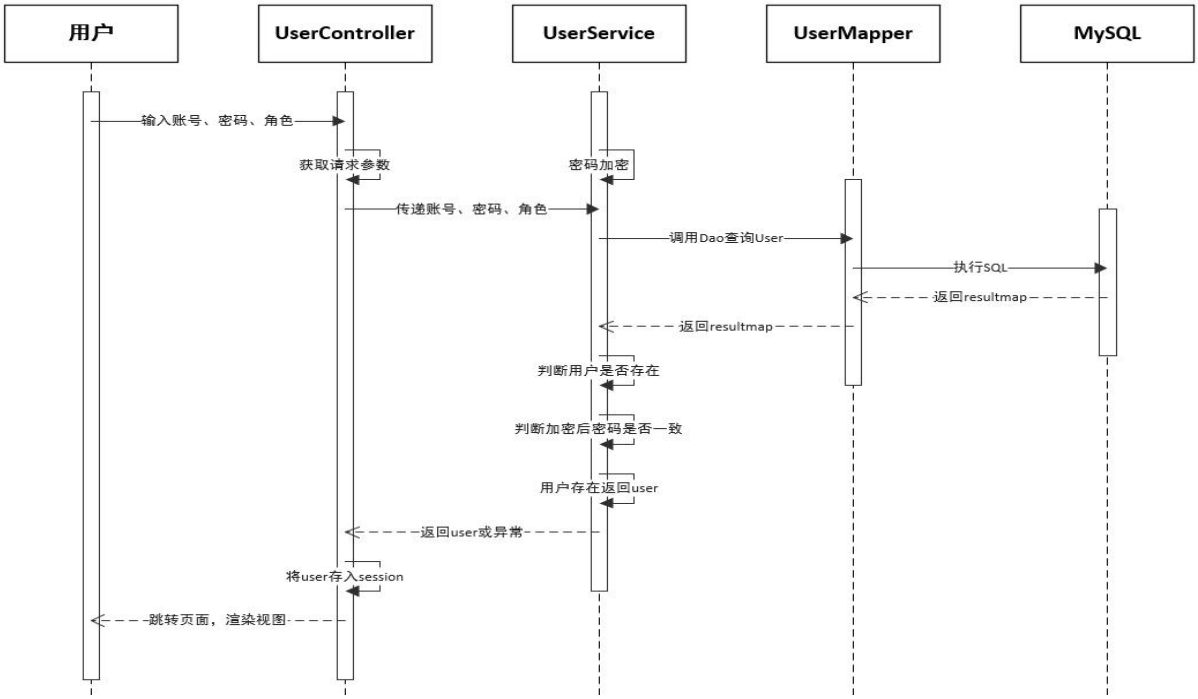


图 5-9 用户登录时序图

5.2.2 测试用例管理

测试用例管理首先包括添加、修改、删除、查询基本功能，具体时序图如 5-10 所示：

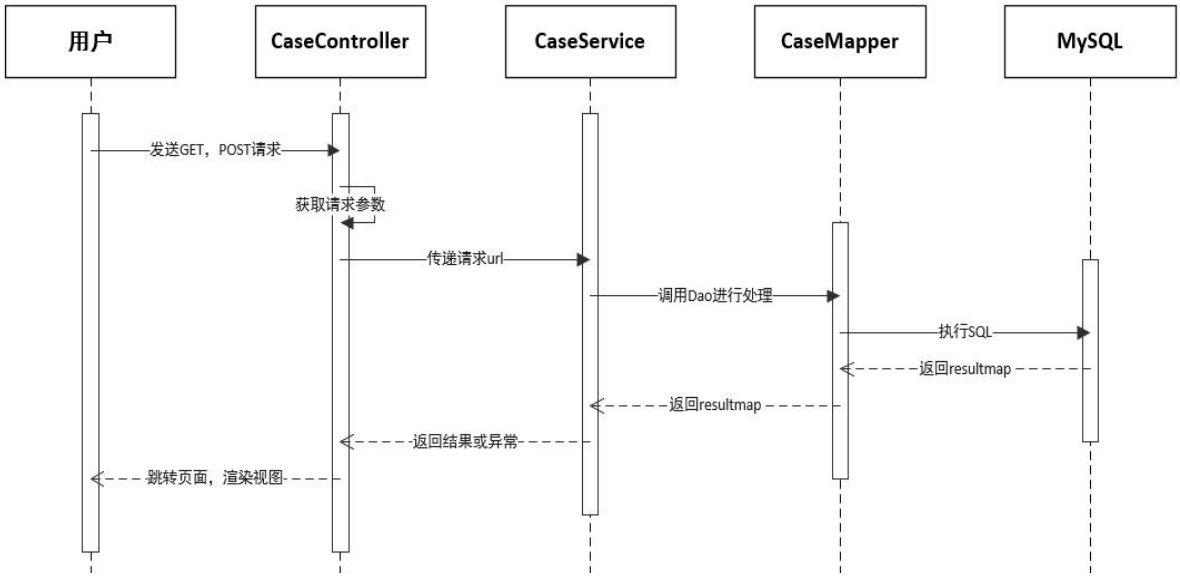


图 5-10 测试用例增删改查时序图

测试用例添加界面如图 5-11 所示：

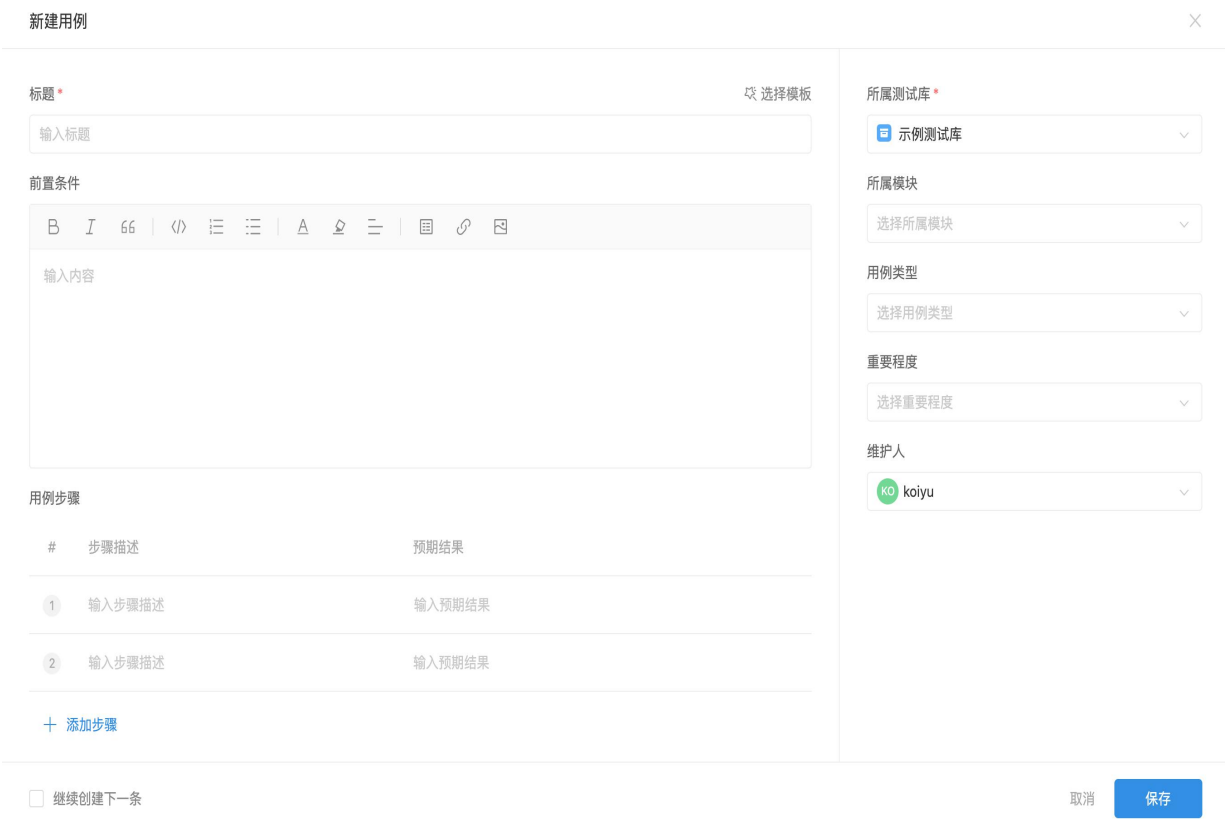


图 5-11 测试用例添加图

测试用例修改界面如图 5-12 所示：

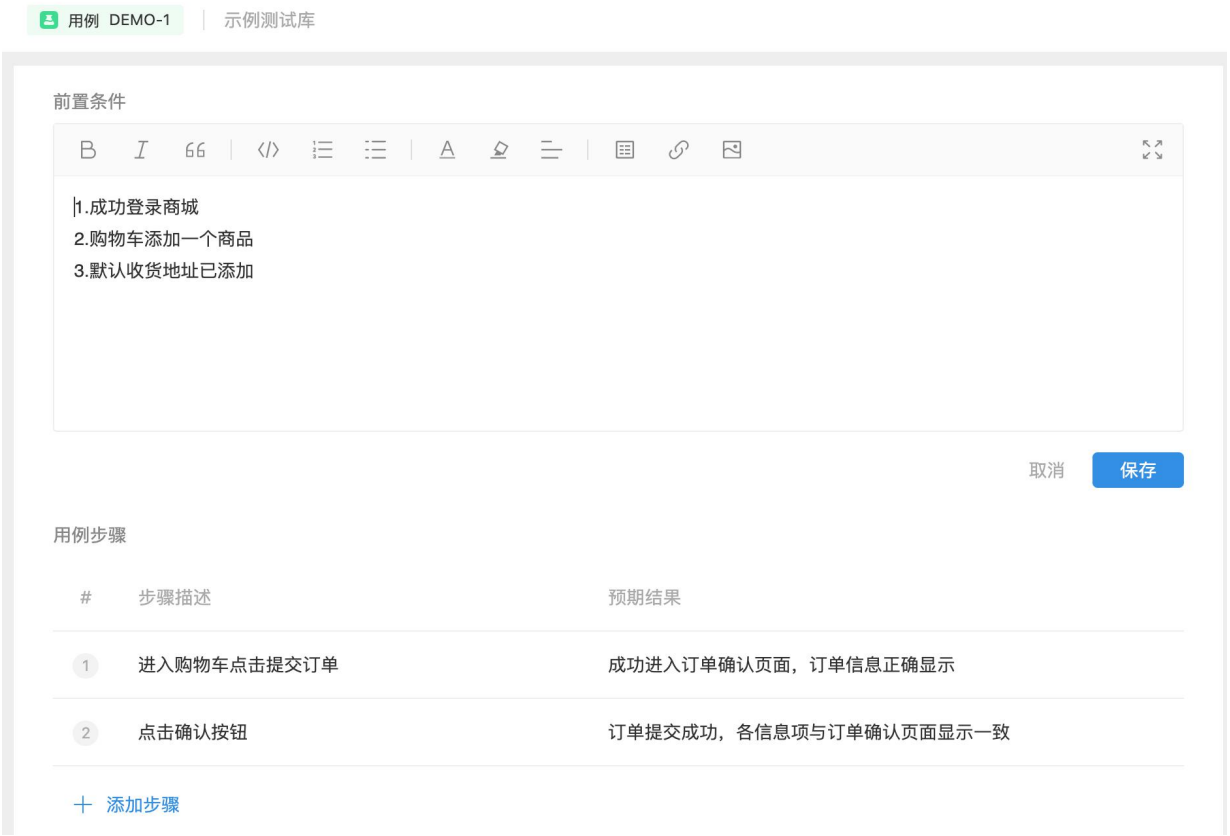


图 5-12 测试用例修改图

测试用例删除界面如图 5-13 所示：

图 5-13 测试用例删除图

测试用例查询界面如图 5-14 所示：

图 5-14 测试用例查询图

除以上功能外，测试用例管理还包括导入、导出功能，使用 Python 封装好的 ExcelWriter 库进行 Excel 读写功能，其流程图如图 5-15 所示：

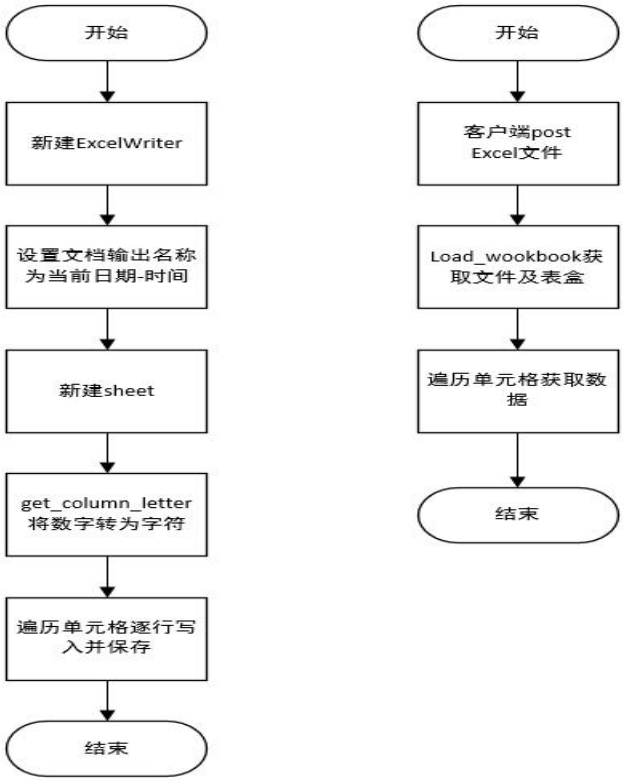
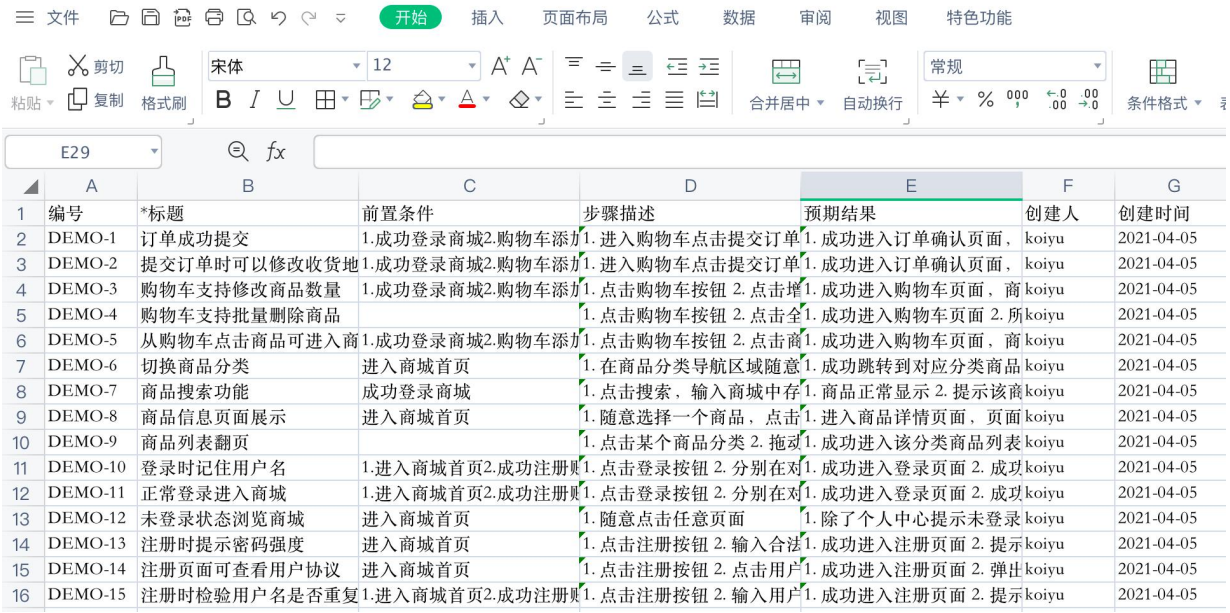


图 5-15 测试用例导入（左）、导出（右）流程图

测试用例导入界面如图 5-14 所示：

图 5-14 测试用例导入图

测试用例导出界面如图 5-15 所示：



	A	B	C	D	E	F	G
1	编号	*标题	前置条件	步骤描述	预期结果	创建人	创建时间
2	DEMO-1	订单成功提交	1.成功登录商城2.购物车添加	1. 进入购物车点击提交订单	1. 成功进入订单确认页面，	koiyu	2021-04-05
3	DEMO-2	提交订单时可以修改收货地	1.成功登录商城2.购物车添加	1. 进入购物车点击提交订单	1. 成功进入订单确认页面，	koiyu	2021-04-05
4	DEMO-3	购物车支持修改商品数量	1.成功登录商城2.购物车添加	1. 点击购物车按钮 2. 点击增	1. 成功进入购物车页面，商	koiyu	2021-04-05
5	DEMO-4	购物车支持批量删除商品		1. 点击购物车按钮 2. 点击全	1. 成功进入购物车页面 2. 所	koiyu	2021-04-05
6	DEMO-5	从购物车点击商品可进入商	1.成功登录商城2.购物车添加	1. 点击购物车按钮 2. 点击商	1. 成功进入购物车页面，商	koiyu	2021-04-05
7	DEMO-6	切换商品分类	进入商城首页	1. 在商品分类导航区域随意	1. 成功跳转到对应分类商品	koiyu	2021-04-05
8	DEMO-7	商品搜索功能	成功登录商城	1. 点击搜索，输入商城中存	1. 商品正常显示 2. 提示该商	koiyu	2021-04-05
9	DEMO-8	商品信息页面展示	进入商城首页	1. 随意选择一个商品，点击	1. 进入商品详情页面，页面	koiyu	2021-04-05
10	DEMO-9	商品列表翻页		1. 点击某个商品分类 2. 拖动	1. 成功进入该分类商品列表	koiyu	2021-04-05
11	DEMO-10	登录时记住用户名	1.进入商城首页2.成功注册账	1. 点击登录按钮 2. 分别在对	1. 成功进入登录页面 2. 成功	koiyu	2021-04-05
12	DEMO-11	正常登录进入商城	1.进入商城首页2.成功注册账	1. 点击登录按钮 2. 分别在对	1. 成功进入登录页面 2. 成功	koiyu	2021-04-05
13	DEMO-12	未登录状态浏览商城	进入商城首页	1. 随意点击任意页面	1. 除了个人中心提示未登录	koiyu	2021-04-05
14	DEMO-13	注册时提示密码强度	进入商城首页	1. 点击注册按钮 2. 输入合法	1. 成功进入注册页面 2. 提示	koiyu	2021-04-05
15	DEMO-14	注册页面可查看用户协议	进入商城首页	1. 点击注册按钮 2. 点击用户	1. 成功进入注册页面 2. 弹出	koiyu	2021-04-05
16	DEMO-15	注册时检验用户名是否重复	1.进入商城首页2.成功注册账	1. 点击注册按钮 2. 输入用户	1. 成功进入注册页面 2. 提示	koiyu	2021-04-05

图 5-15 测试用例导出图

5.2.3 自动化测试框架管理

自动化测试框架管理主要实现代码管理、分支覆盖率统计和自动化测试用例执行报表生成，具体内容如下：

（1）代码管理目前复用 GitHub 界面，点击代码管理，跳转至 GitHub 网址，如图 5-16 所示；

（2）覆盖率统计当前报表为作者代码三个分支插入 JaCoCo 插件后得出的结果，如图 5-17 所示；

（3）自动化测试用例执行报告目前主要通过手工执行单元测试用例，数据来源暂时为作者构造，如图 5-18 所示。

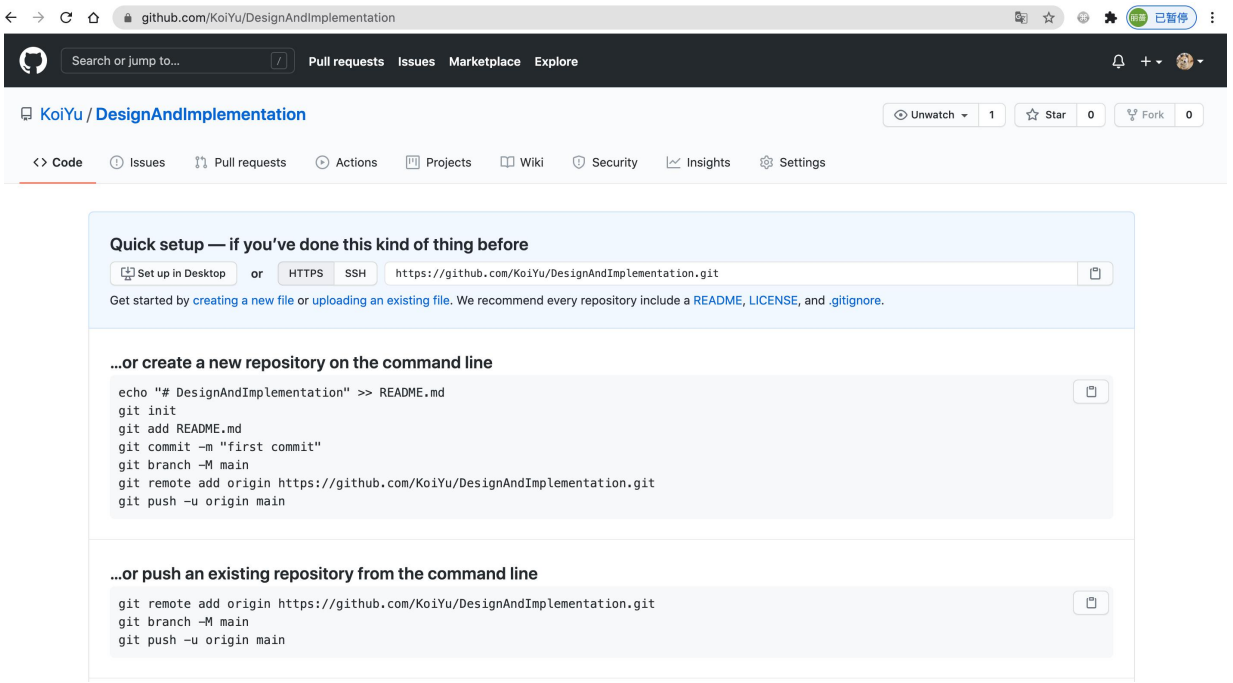


图 5-16 代码管理界面图

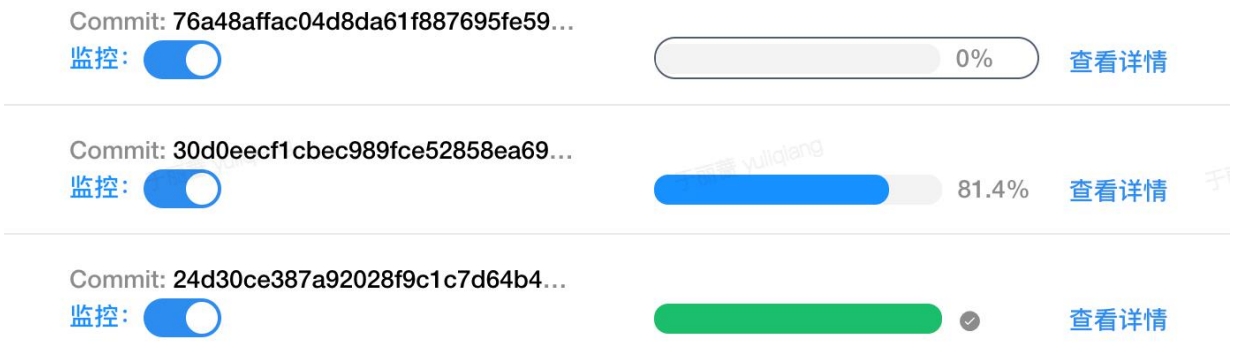


图 5-17 覆盖率统计界面图

任务ID	状态	通过率	触发方式	测试计划	创建时间	执行时长
2056814	完成-未通过	通过:14 ; 失败:2 ; 跳过:0	cronjob	全量测试计划	2021-05-05 11:00:01	2 分 41 秒
2037540	完成-未通过	通过:14 ; 失败:3 ; 跳过:0	cronjob	全量测试计划	2021-05-04 11:00:01	2 分 41 秒
2018256	完成-未通过	通过:14 ; 失败:3 ; 跳过:0	cronjob	全量测试计划	2021-05-03 11:00:01	2 分 36 秒
1998917	完成-未通过	通过:14 ; 失败:2 ; 跳过:0	cronjob	全量测试计划	2021-05-02 11:00:01	2 分 30 秒
1979662	完成-未通过	通过:14 ; 失败:2 ; 跳过:0	cronjob	全量测试计划	2021-05-01 11:00:01	2 分 31 秒
1959792	完成-未通过	通过:14 ; 失败:2 ; 跳过:0	cronjob	全量测试计划	2021-04-30 11:00:01	2 分 35 秒
1939415	完成-未通过	通过:14 ; 失败:3 ; 跳过:0	cronjob	全量测试计划	2021-04-29 11:00:01	2 分 30 秒
1917810	完成-未通过	通过:14 ; 失败:2 ; 跳过:0	cronjob	全量测试计划	2021-04-28 11:00:01	2 分 32 秒

图 5-17 自动化测试报告界面图

5.2.4 测试问题处理

平台实现测试用例与缺陷报告的关联，在此介绍一下缺陷问题处理流程，流程图如图 5-18：

（1）测试人员执行测试用例时发现问题，新建缺陷报告，并将测试用例链接复制到缺陷报告中，缺陷报告建立后将链接复制到测试用例中，此时缺陷报告状态为新提交；

（2）开发人员查收缺陷报告，并根据测试用例链接查看复现步骤，以便排查、修复问题，确认问题已解决后将缺陷报告状态置为已修复；

（3）测试人员进行已修复问题的回归验证，验证不通过将缺陷状态置为重新打开，待开发人员重复流程（2），验证通过则将缺陷问题置为已发布，意为已解决，至此缺陷问题处理流程结束。

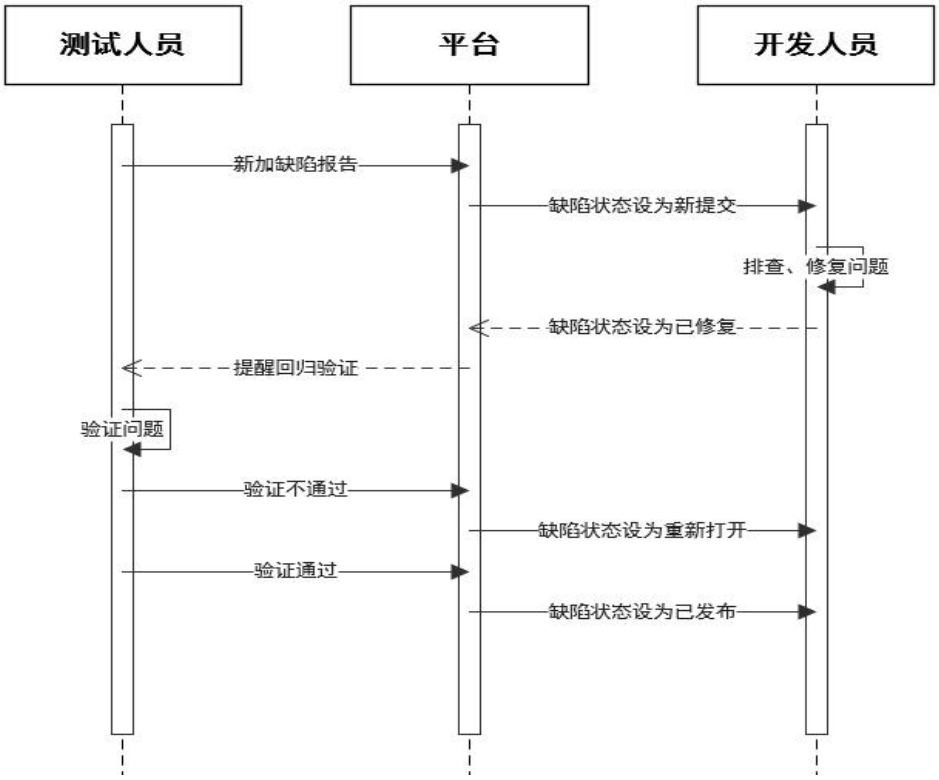


图 5-18 缺陷问题处理流程图

5.3 平台核心算法层设计与实现

本平台核心问题在于测试用例的自动生成，当前可根据对需求文档进行分词后，根据已有测试用例进行推荐生成，后续将实现根据基础测试理论进行自动生成。

5.3.1 分词算法

Python 中文分词软件中的“结巴”分词，具体实现如图 5-19 所示：

```
import  os
import  jieba
class  Segmentation:
      files  =  []
      def  __init__(self):
          self.files  =  os.listdir("contents")
```

```
    def segment(self):
        count = 0
        for file in self.files:
            count+=1
            if count == 2:
                break
            f = open("contents" + "/" + file, encoding='utf-8')
            content = f.readline()
            seg_list = jieba.cut(content)    # 默认是精确模式
            print(", ".join(seg_list))
if __name__ == '__main__':
    segmentation = Segmentation()
    segmentation.segment()
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\ADMINI~1\AppData\Local\Temp\jieba.cache
购物车, 支持, 添加, 、, 删除, 商品, , , 商品信息, 页面, 显示, 列表, , , 正常, 登录, 下, 可, 浏览, 商城
Loading model cost 1.587 seconds.
Prefix dict has been built successfully.

Process finished with exit code 0
```

图 5-19 分词结果图

5.3.2 推荐算法

在进行生成测试用例推荐文本之前，需要使用 Apriori 算法对需求文本进行数据挖掘，得到关联规则，并存入数据库[15]。实现结果如图 5-20 所示：

```
public CommonReturnType apriori(){
    List<Case> cases = caseMapper.list();
    List<ProductDoc> prds = productDocMapper.list();
    Map<Case,Integer> support = getSupport(cases,prds,2);
    //现在这个 Map 里存放的即为 L1
    List<Rule>L1 = new ArrayList<>();
    for (Case case:support.keySet()){
        Rule rule = new Rule();
        rule.setAtts(case.getName());
        int occ=support.get(case);
        if (occ==0)continue;
        rule.setOccurence(occ);
        rule.setSize(1);
        L1.add(rule);
        ruleMapper.insert(rule);
    }
    dfs(L1,cases,prds,2);
    return CommonReturnType.create("sucess");
}
private void dfs(List<Rule>lk,List<Case>cases,List<ProductDoc>prds,int
miniSupport){
```

```
        ArrayList<Rule> lk1 = getLkPlus1(lk,cases,prds,miniSupport);
        if (lk1==null)return;
        for (Rule rule:lk1){
              ruleMapper.insert(rule);
        }
        dfs(lk1,cases,prds,miniSupport);
}
```

K12 fx

	A	B	C	D	E
1	编号	*标题	前置条件	步骤描述	预期结果
2	DEMO-3	购物车支持修改商品数量	1.成功登录商城2.购物车添加	1. 点击购物车按钮 2. 点击	1. 成功进入购物车页面，商品
3	DEMO-4	购物车支持批量删除商品		1. 点击购物车按钮 2. 点击	1. 成功进入购物车页面 2. 用
4	DEMO-5	从购物车点击商品可进入商品	1.成功登录商城2.购物车添加	1. 点击购物车按钮 2. 点击	1. 成功进入购物车页面，商品
5	DEMO-6	切换商品分类	进入商城首页	1. 在商品分类导航区域随意	1. 成功跳转到对应分类商品列
6	DEMO-7	商品搜索功能	成功登录商城	1. 点击搜索，输入商城中存	1. 商品正常显示 2. 提示该商
7	DEMO-8	商品信息页面展示	进入商城首页	1. 随意选择一个商品，点击	1. 进入商品详情页面，页面信
8	DEMO-9	商品列表翻页		1. 点击某个商品分类 2. 拖	1. 成功进入该分类商品列表
9	DEMO-10	登录时记住用户名	1.进入商城首页2.成功注册账	1. 点击登录按钮 2. 分别在	1. 成功进入登录页面 2. 成功
10	DEMO-11	正常登录进入商城	1.进入商城首页2.成功注册账	1. 点击登录按钮 2. 分别在	1. 成功进入登录页面 2. 成功
11	DEMO-12	未登录状态浏览商城	进入商城首页	1. 随意点击任意页面	1. 除了个人中心提示未登录，

图 5-19 测试用例推荐结果图

5.4 本章小结

本章主要从一些界面、功能逻辑、核心算法的详细设计入手，介绍了系统中一些关键逻辑和关键技术的实现过程，这些关键技术共同构成了整个系统的核心。到本章为止系统的全部业务基本实，接下来，本研究将会对系统开展功能相关测试。

6 测试

6.1 测试目的

在完成平台的搭建之后，需要验证平台各项功能的正确性和可靠性等，保证功能正常运行等，且应当符合软件工程准则，有良好的使用体验。

6.2 测试方法

鉴于平台性质，测试部分主要采用功能测试。功能测试主要根据产品特性、用户需求和基本要求，判断系统的可执行性和可操作性是否满足设计需求，主要侧重于业务功能和业务规则。

6.3 测试工具及环境

为了验证本文设计实现是否符合用户需求，本章对系统的各个功能模块进行了功能测试，并验证了系统在并发情况下的基本性能。具体测试环境如下：

(1) 硬件环境:

●服务器：本地主机一台

●CPU：Intel i7 / 2.6GHZ

●RAM：16G

●硬盘：512G

(2) 软件环境:

●操作系统：Windows10

●测试工具：Jmeter，IntelliJ IDEA，Google Chrome

6.4 功能测试

本文功能测试主要包括 UI 测试，以及各个功能模块的测试。

6.4.1 UI 测试

系统通过 UI 界面向用户提供可视化操作，因此需要通过 UI 测试来检验用户与系统之间的交互情况。下面测试以菜单、窗口和图形为例进行说明。

（1）菜单

针对于菜单的 UI 测试主要包含菜单与实际提供的功能相符，菜单上无错别字且语义表达清晰准确，快捷键或热键与操作可以互相对应。

（2）窗口

针对于窗口测试，主要包括窗口大小合适、外观美观、布局合理。窗口不同分辨率设置下可以正常显示，窗口放大缩小后内容可以正常显示，窗口中图标与功能一一对应，文字描述准确且无错别字。

（3）图形

图形测试内容包括图片、边框、字体、颜色、按钮等，确保色调和谐，风格一致，提高用户体验性。

6.4.2 用户管理模块功能测试

用户管理模块主要包括用户的添加、修改、删除、查询、权限分配等几项功能，本小节对于每一项功能都进行了响应测试，整体测试情况如下表 7-1 所示。

表 7-1 用户管理模块功能测试结果

用例编号	测试输入	预期输出	实际输出	是否通过
A01	添加用户，信息完整	添加成功，提示新增成功并显示新增用户	添加成功，提示新增成功并显示新增用户	是
A02	添加用户，信息错误	不可添加，错误提示	不可添加，错误提示	是
A03	添加用户，信息缺失	不可添加，错误提示	不可添加，错误提示	是
A04	修改用户信息，信息正确，点击保存	保存成功，提示修改成功并显示修改信息	修改成功，提示修改成功并显示修改信息	是
A05	修改用户信息，信息错误，点击保存	保存失败，错误提示	保存失败，错误提示	是
A06	通过姓名查询用户	查询成功，非唯一索引可查询多条	查询成功，查询多条姓名相同用户	是
A07	通过手机号查询用户	查询成功，结果正确	查询成功，结果正确	是
A08	通过部门查询用户	查询成功，非唯一索引可查询多条	查询成功，查询多条部门相同用户	是
A09	通过角色查询用户	查询成功，非唯一索引可查询多条	查询成功，查询多条角色相同用户	是
A10	通过不存在信息查询用户	查询成功，查询结果为空	查询成功，查询结果为空	是
A11	组合条件查询用户	查询成功，结果正确	查询成功，结果正确	是
A12	不设置查询条件查询	查询成功，显示所有用户信息	查询成功，显示所有用户信息	是
A13	删除一条用户记录，删除弹窗点击确认	删除成功，提示删除成功并显示用户列表	删除成功，提示删除成功并显示用户列表	
A14	删除多条用户记录，删除弹窗点击确认	删除成功，提示删除成功并显示用户列表	删除成功，提示删除成功并显示用户列表	是
A15	删除弹窗点击取消	删除失败，提示取消操作并显示用户列表	删除成功，提示取消操作并显示用户列表	是
A16	分配角色为管理员，登录账号	登录成功，显示管理员界面	登录成功，显示管理员界面	是
A17	分配角色为测试人员，登录账号	登录成功，显示测试人员界面	登录成功，显示测试人员界面	是
A18	分配角色为开发人员，登录账号	登录成功，显示开发人员界面	登录成功，显示开发人员界面	是

6.4.3 测试用例管理模块功能测试

测试用例管理模块主要包括测试用例的添加、修改、删除、查询、导入、导出等几项功能，本小节对于每一项功能都进行了响应测试，整体测试情况如下表 7-12 所示。

表 7-2　测试用例管理模块功能测试结果

用例编号	测试输入	预期输出	实际输出	是否通过
B01	添加测试项目	弹窗填写项目信息	弹窗填写项目信息	是
B02	填写测试项目信息，信息完整，点击保存	添加成功，提示新增成功并显示新增项目	添加成功，提示新增成功并显示新增项目	是
B03	填写测试项目信息，信息缺失，点击保存	不可添加，错误提示	不可添加，错误提示	是
B04	点击测试用例管理，查看测试项目	菜单下拉，显示测试项目	菜单下拉，显示测试项目	是
B05	右键点击测试项目，修改测试项目	弹窗填写项目信息	弹窗填写项目信息	是

B06	修改项目信息，信息正确，点击保存	保存成功，提示修改成功并显示修改信息	修改成功，提示修改成功并显示修改信息	是
B07	修改项目信息，信息错误，点击保存	保存失败，错误提示	保存失败，错误提示	是
B08	右键点击测试项目，删除测试项目	弹窗提示操作确认	弹窗提示操作确认	是
B09	删除测试项目，删除弹窗点击确认	删除成功，提示删除成功并显示项目列表	删除成功，提示删除成功并显示项目列表	是
B10	删除测试项目，删除弹窗点击取消	删除失败，提示取消操作并显示项目列表	删除成功，提示取消操作并显示项目列表	是
B11	添加测试用例，信息完整	添加成功，提示新增成功并显示新增用例	添加成功，提示新增成功并显示新增用例	是
B12	添加测试用例，信息缺失	不可添加，错误提示	不可添加，错误提示	是
B13	自动添加测试用例，导入需求文档	导入成功，提示导入成功，文件生成中	导入成功，提示导入成功，文件生成中	是
B14	下载并查看自动生成测试用例文档	文档下载成功，显示推荐测试用例	文档下载成功，显示推荐测试用例	是
B15	修改用例信息，信息正确，点击保存	保存成功，提示修改成功并显示修改信息	修改成功，提示修改成功并显示修改信息	是
B16	修改用例信息，信息错误，点击保存	保存失败，错误提示	保存失败，错误提示	是
B17	通过用例标题模糊查询测试用例	查询成功，非唯一索引可查询多条，查询结果正确	查询成功，非唯一索引可查询多条，查询结果正确	是
B18	通过创建人查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条创建人相同用例	是
B17	通过执行人查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条执行人相同用例	是
B15	通过用例类型查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条用例类型相同用例	是
B16	通过重要程度查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条重要程度相同用例	
B17	通过用例状态查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条用例状态相同用例	是
B18	通过执行结果查询测试用例	查询成功，非唯一索引可查询多条	查询成功，查询多条执行结果相同用例	
B18	通过组合条件查询测试用例	查询成功，结果正确	查询成功，结果正确	是
B17	不设置查询条件查询	查询成功，显示所有用例信息	查询成功，显示所有用例信息	
B15	删除一条用例记录，删除弹窗点击确认	删除成功，提示删除成功并显示用例列表	删除成功，提示删除成功并显示用例列表	是
B16	删除多条用例记录，删除弹窗点击确认	删除成功，提示删除成功并显示用例列表	删除成功，提示删除成功并显示用例列表	是
B17	删除弹窗点击取消	删除失败，提示取消操作并显示用例列表	删除成功，提示取消操作并显示用例列表	是
B18	点击测试用例管理，查看测试用例	显示所有测试用例	显示所有测试用例	是
B19	点击某个测试项目，查看测试用例	显示该项目下所有测试用例	显示该项目下所有测试用例	是

B20	选中测试用例，点击导出	提示导出成功，请下载用例文档	提示导出成功，请下载用例文档	是
B21	下载用例文档并查看	下载成功，显示用例	下载成功，显示用例	是
B22	点击导入测试用例	弹窗提示下载导入用例模板	弹窗提示下载导入用例模板	是
B23	填写模板，信息完整，点击导入	导入成功，提示导入成功，显示用例信息	导入成功，提示导入成功，显示用例信息	是
B24	填写模板，信息缺失，点击导入	导入失败，提示错误	导入失败，提示错误	是
B25	点击缺陷报告链接	跳转缺陷报告页面	跳转缺陷报告页面	是
B26	点击测试用例分享	复制测试用例链接	复制测试用例链接	是

6.4.4 缺陷报告管理模块功能测试

缺陷报告管理模块主要包括缺陷报告的添加、修改、删除、查询、导入、导出等几项功能，本小节对于每一项功能都进行了响应测试，整体测试情况如下表 7-12 所示。

表 7-2 测试用例管理模块功能测试结果

用例编号	测试输入	预期输出	实际输出	是否通过
C01	添加缺陷报告，信息完整，点击保存	添加成功，提示新增成功并显示新增项目	添加成功，提示新增成功并显示新增项目	是
C02	添加缺陷报告，信息缺失，点击保存	不可添加，错误提示	不可添加，错误提示	是
C03	修改缺陷报告信息，信息正确，点击保存	保存成功，提示修改成功并显示修改信息	修改成功，提示修改成功并显示修改信息	是
C04	修改缺陷报告信息，信息错误，点击保存	保存失败，错误提示	保存失败，错误提示	是
C05	通过问题描述模糊查询缺陷报告	查询成功，非唯一索引可查询多条，查询结果正确	查询成功，非唯一索引可查询多条，查询结果正确	是
C06	通过发现人查询缺陷报告	查询成功，非唯一索引可查询多条	查询成功，查询多条发现人相同报告	是
C07	通过跟进人查询缺陷报告	查询成功，非唯一索引可查询多条	查询成功，查询多条跟进人相同报告	是
C08	通过缺陷状态查询缺陷报告	查询成功，非唯一索引可查询多条	查询成功，查询多条缺陷状态相同报告	是
C09	通过优先级查询缺陷报告	查询成功，非唯一索引可查询多条	查询成功，查询多条优先级相同报告	是
C10	通过执行结果查询缺陷报告	查询成功，非唯一索引可查询多条	查询成功，查询多条执行结果相同报告	是
C11	通过组合条件查询缺陷报告	查询成功，结果正确	查询成功，结果正确	是
C12	不设置查询条件查询缺陷报告	查询成功，显示所有报告信息	查询成功，显示所有报告信息	是
C13	点击缺陷报告分享	复制缺陷报告链接	复制缺陷报告链接	是
C14	点击测试用例链接	跳转测试用例页面	跳转测试用例页面	是
C15	点击缺陷报告管理，查看缺陷报告	显示所有缺陷报告	显示所有缺陷报告	是
C17	点击某个测试项目，查看缺陷报告	显示该项目下所有缺陷报告	显示该项目下所有缺陷报告	是

6.4.5 自动化测试框架管理模块功能测试

测试用例管理模块主要包括缺陷报告的添加、修改、删除、查询、导入、导出等几项功能，本小节对于每一项功能都进行了响应测试，整体测试情况如下表 7-12 所示。

表 7-2 测试用例管理模块功能测试结果

用例编号	测试输入	预期输出	实际输出	是否通过
D01	点击代码管理	跳转 GitHub 页面	跳转 GitHub 页面	是
D02	点击覆盖率统计	跳转统计页面，显示使用插件代码覆盖率	跳转统计页面，显示使用插件代码覆盖率	是
D03	点击统计报表	显示测试报告和其他报表统计	显示测试报告和其他报表统计	是

测试之后可以发现，预期输出与实际输出结果相同，因此测试均为通过。

6.5 本章小结

本章主要是对系统进行了功能测试。经过了上述测试之后，我们可以验证系统的实际功能和性能符合最初的需求，可以正常投入使用。至此为止整个系统的构造完成。在最后的结论部分将会对整个系统进行全面分析。

结论

本平台基于 B/S 架构，根据软件工程开发思想完成了全部的开发任务。本次开发搭建了一款测试事务管理平台，为测试人员和开发人员工作流程和项目闭环提供了很好的便利。

开发流程上，按照软件生命周期流程分为可行性研究、需求分析、概要设计、详细设计、编码、测试等步骤。开发过程中，逐步构建完成整个平台，并对平台的功能和性能进行测试。平台主要基于软件工程中常用的增量模型进行开发，增量 1—用户管理模块开发，增量 2—测试用例管理模块基础框架开发，增量 3—测试用例自动生成功能开发，增量 4—自动化测试框架管理模块基础框架开发，增量 5—测试用例关联测试报告功能开发即缺陷报告管理模块开发。

功能上，平台创新性地实现自动生成测试用例，提高测试效率；并设计了一款基于测试点集的推荐算法，以内容过滤为核心思想，提供可用测试用例供测试人员选择。同时，平台实现了项目管理、测试用例管理、缺陷报告管理全流程跟进，便于测试人员和开发人员敏捷、高效合作。

架构上，平台以 B/S 三层架构作为基础，将平台的服务划分为界面表现层、业务逻辑层和数据三大模块。界面表现层为用户提供了可视化的操作界面；业务逻辑层实现各个业务模块的功能需求，降低业务之间的耦合性；数据层主要提供数据的同步和读写操作。三者各司其职，提高整个平台的灵活性和易用性。

综上，本平台采用了规范的软件开发方法以保证软件开发过程的完整性、正确性和可度量性；经过系统测试验证，所开发的平台在业务内容上能够满足用户需求；所采用的三层架构有利于后续软件维护与升级。

系统的后续完善将从以下方面展开：

（1）进一步提高平台的操作易用性，增强用户体验。

（2）根据实际使用情况进行开发流程的进一步细化，提高软件生命全周期的可维护性。

（3）进一步完善测试用例自动生成的准确性与便捷性，尽可能精准的提供可供选择的推荐内容，且尽可能直接的生成准确可用的测试用例。