软件需求规格说明书

基于Flask的深度学习自动化部署系统

|  |  |  |  |
| --- | --- | --- | --- |
| 编写 | 许京爽，张文斌，李坤浩，张竹君，崔昕宇，张利鹏，聂磊 | 日期 | 2020年3月30日 |
| 校对 | 聂磊 | 日期 | 2020年4月10日 |

北京航空航天大学 计算机学院

软件工程综合实验课 C组

二〇二〇年四月十日

**修订记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 | 审核人 |
| 1.0.0 | 2020.3.25 |  | 完成第一版 | 许京爽，张文斌，聂磊 | 许京爽 |
| 1.0.1 | 2020.3.27 | 3 | 添加业务需求与系统的关系图 | 聂磊 | 许京爽 |
| 2.0.0 | 2020.4.2 | 1,2,3,4,5 | 文章结构规范化重构 | 许京爽，张文斌，李坤浩，张竹君，崔昕宇，张利鹏，聂磊 | 聂磊 |
| 2.0.1 | 2020.4.2 | 3 | 更改外部接口需求的MySQL部分 | 崔昕宇 | 聂磊 |
| 2.0.2 | 2020.4.2 | 3 | 删除原有3.1.10章节：将模型部署到服务器 | 张利鹏 | 聂磊 |
| 2.0.3 | 2020.4.3 | 3 | 新增章节3.1.14与3.1.15：暂停与重启运行中的实例 | 张文斌 | 聂磊 |
| 2.0.4 | 2020.4.4 | 参考文献 | 参考文献单列一章 | 聂磊 | 许京爽 |
| 2.1.0 | 2020.4.9 |  | 根据4月8日的评审结果进行修订 | 聂磊 | 许京爽 |
| 2.1.1 | 2020.4.10 | 3.1 | 去掉了模型管理中的暂停已部署服务 | 张竹君 | 聂磊 |
| 2.1.2 | 2020.4.10 | 3.1 | 更新用例图 | 张竹君 | 聂磊 |

**目录**

[1 简介 1](#_Toc37419979)

[1.1 目的 1](#_Toc37419980)

[1.2 背景 1](#_Toc37419981)

[2 总体概述 1](#_Toc37419982)

[2.1 软件概述 1](#_Toc37419983)

[2.2 软件功能 2](#_Toc37419984)

[2.3 用户特征 2](#_Toc37419985)

[2.4 软件依赖关系 3](#_Toc37419986)

[3 具体需求 3](#_Toc37419987)

[3.1 功能需求 3](#_Toc37419988)

[3.1.1 账户管理：用户注册 3](#_Toc37419989)

[3.1.2 账户管理：用户登录 4](#_Toc37419990)

[3.1.3 账户管理：用户登出 5](#_Toc37419991)

[3.1.4 项目管理：新建项目 5](#_Toc37419992)

[3.1.5 项目管理：查看项目 6](#_Toc37419993)

[3.1.6 项目管理：删除项目 6](#_Toc37419994)

[3.1.7 项目管理：更新项目 6](#_Toc37419995)

[3.1.8 模型管理：在项目中导入模型 7](#_Toc37419996)

[3.1.9 模型管理：对模型部署的参数进行设置 8](#_Toc37419997)

[3.1.10 模型管理：查看某一项目下所有模型及相关配置 9](#_Toc37419998)

[3.1.11 模型管理：删除模型 10](#_Toc37419999)

[3.1.12 实例管理：启动实例 11](#_Toc37420000)

[3.1.13 实例管理：删除实例 12](#_Toc37420001)

[3.1.14 实例管理：暂停运行中实例 13](#_Toc37420002)

[3.1.15 实例管理：恢复暂停中实例 14](#_Toc37420003)

[3.1.16 使用已部署的模型 15](#_Toc37420004)

[3.2 性能需求 15](#_Toc37420005)

[3.2.1 同时运行模型实例数 15](#_Toc37420006)

[3.2.2 网站界面并发访问能力 15](#_Toc37420007)

[3.2.3 REST API并发访问能力 15](#_Toc37420008)

[3.3 外部接口需求 15](#_Toc37420009)

[3.3.1 用户接口 15](#_Toc37420010)

[3.3.2 软件接口 16](#_Toc37420011)

[3.3.3 硬件接口 17](#_Toc37420012)

[3.3.4 通讯接口 17](#_Toc37420013)

[4 总体设计约束 17](#_Toc37420014)

[4.1 标准符合性 17](#_Toc37420015)

[4.2 硬件约束 18](#_Toc37420016)

[4.3 软件约束 18](#_Toc37420017)

[5 软件质量特征 18](#_Toc37420018)

[5.1兼容性 18](#_Toc37420019)

[5.2 可扩展性 18](#_Toc37420020)

[5.3 容灾性 18](#_Toc37420021)

[附录 A Flask框架分析 19](#_Toc37420022)

[A.1 框架介绍 19](#_Toc37420023)

[A.2 框架核心 19](#_Toc37420024)

[A.3 Flask通用扩展包 19](#_Toc37420025)

[参考文献 20](#_Toc37420026)

基于Flask的深度学习自动化部署系统

软件需求规格说明书

**关键词：**深度学习，软件即服务（SaaS）

**摘要：**本文描述了2020春季学期《软件工程综合实验》课程中，C组成员开发的深度学习自动化部署系统的需求规格。

**专业术语清单：**

|  |  |
| --- | --- |
| **专业术语** | **中文解释** |
| Web应用框架 | Web应用框架是一种开发框架，用来支持动态网站、网络应用程序及网络服务的开发，有助于减轻网页开发时共通性活动的工作负荷。 |
| Python | Python是一种跨平台的计算机程序设计语言，具有面向对象和动态类型的特性。Python最初被设计用于编写自动化脚本，随着版本的不断更新和语言新功能的添加，越多被用于独立的、大型项目的开发。 |
| WSGI | Web服务器网关接口（英文全称：Python Web Server Gateway Interface）是为Python语言定义的Web服务器和Web应用程序或框架之间的一种简单而通用的接口。 |
| 深度学习 | 深度学习是机器学习领域中一个新的研究方向，通过学习样本数据的内在规律和表示层次，让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。 |
| 模型部署 | 将训练完毕的深度学习模型放置在特定硬件平台上，辅以合适的软件环境，使该模型能够被用户使用。 |
| HTTP | 超文本传输协议（英文全称：HyperText Transfer Protocol）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。 |
| REST API | 表现层状态转换（英文全称：Representational State Transfer）是一种万维网软件架构风格，目的是便于不同软件/程序在网络中互相传递信息。表现层状态转换是根基于超文本传输协议之上而确定的一组约束和属性，是一种设计提供万维网络服务的软件构建风格。 |

# 1 简介

## 1.1 目的

本文档是C组开发项目 ***基于Flask的深度学习自动化部署系统*** 的需求规格说明书，通过对项目预期功能的评估，在明确定义和用户需求的基础上，进一步确定软件的功能需求、性能需求，以及软件对运行环境的要求。

本轮需求分析过程以Flask开源框架及相关参考资料为输入，分析软件设计需求，结合软件工程综合实验具体要求，输出软件需求规格说明书。

本文档是为软件设计人员、开发人员、测试人员、项目验收方编写的。可作为后续软件架构设计的依据，软件开发过程的参考材料，测试过程的评价指标，以及项目验收达标的依据。

## 1.2 背景

Flask是Python常用的Web开发框架，因其具有很多优点而受到广泛的使用。它非常的灵活、轻便、安全且容易上手，能够很好地结合MVC模式进行开发，具有高度定制性，用户可以根据自己的需求来添加相应的功能，在保持核心结构简单的同时，实现功能的丰富与扩展。因其具有强大的插件库，可以让用户实现个性化的网站定制，开发出功能强大的网站。

# 2 总体概述

## 2.1 软件概述

高效的深度学习模型需要通过实际部署才能在生产生活中发挥作用。常用的部署方式是将模型包装成Web服务，向外界提供基于HTTP协议的接口，从而提供各类预测功能和服务。对于算法研究人员而言，将深度学习模型部署为Web服务的学习成本较高，通常需要工程开发人员协助部署，这一定程度上降低了算法研发和实际部署的效率。本软件旨在通过自动化部署深度模型，降低算法研究人员实际部署模型的门槛，提高深度学习模型迭代研发和部署的效率。

项目采用基于Python语言的轻量级Web框架Flask和容器技术作为实际的模型部署工具，自动化部署系统包括了网站前端页面，网站后端模块和模型部署模块。用户通过Web界面完成模型的上传和部署，并在随后通过Web界面获得访问该模型的REST API。在完成模型的部署之后，用户可在需要使用该模型的场景下调用此REST API，把待识别的图片发送至服务器，服务器将通过HTTP响应返回图片的识别结果。本项目的使用方式如图1所示。

调用API

模型上传

**图1 本项目的使用方式示意图**

## 2.2 软件功能

基于Flask的深度学习自动化部署系统实现以下功能：

1. 账户管理：对本系统网站账户的注册、登录、登出；
2. 项目管理：对用户部署项目的创建、修改、删除；
3. 模型管理：上传、配置深度学习模型文件；
4. 在服务器上部署深度学习模型，通过REST API访问运行中的模型实例。

## 2.3 用户特征

本软件系统的用户应当具有深度学习的相关理论和实践知识，能够正确地将待解决的问题以恰当的深度学习模型进行描述，并将训练完毕的深度学习模型上传至本系统。

此外，用户通过本系统完成模型的部署后，模型的使用方应当能够正确地通过REST API访问本软件提供的服务接口，以上传待神经网络分析的数据（例如照片信息），并获取神经网络的输出结果（例如物体识别的结果）。

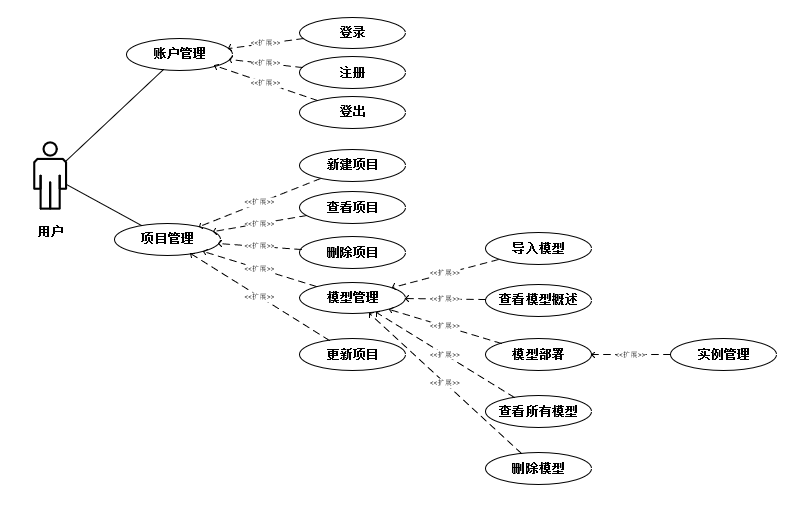
## 软件依赖关系

1. 依赖Flask 1.1.1提供的Web程序框架；
2. 依赖TensorFlow 2.1.0与PyTorch 1.0.2提供的深度学习模型运行框架；
3. 依赖MySQL 5.7提供的数据库接口。

# 3 具体需求

## 功能需求

本软件实现的功能所对应的用例图如图1所示。



**图2 软件用例图**

### 3.1.1 账户管理：用户注册

1. 介绍

用户可在网站注册账号，以获得网站功能的使用权限。

1. 输入

用户从页面输入的用户信息，包括用户名、密码、确认密码和邮箱，用户名需要满足字符长度至少为6、至多为20，密码需要满足字符长度至少为6、至多为30。

1. 处理

用户填写的表单数据由前端传递给后端；

IF ( 输入的用户名在账户数据库中已存在 )

{

注册失败，在页面显示提示信息“该用户名已存在”；

}

ELSE IF( 两次输入的密码不匹配 )

{

注册失败，在页面显示提示信息“两次输入的密码不匹配”；

}

ELSE IF ( 输入的邮箱格式不正确 )

{

注册失败，在页面显示提示信息“邮箱格式不正确”；

}

ELSE ( 输入的邮箱在账户数据库中已存在 )

{

注册失败，在页面显示提示信息“该邮箱已被注册”；

}

将用户名、加密后的密码和邮箱存入账户数据库中，以邮箱为主键；

将注册成功的信息反馈给前端。

1. 输出

若用户输入的信息无效，则判定为注册失败，在前端显示错误提示。若输入的信息有效，注册成功，将用户名、密码和邮箱存入账户数据库中，并在前端显示“注册成功”的提示信息。

### 3.1.2 账户管理：用户登录

1. 介绍

用户通过输入正确的用户名和密码登录其账号，管理其私有的项目和模型信息。

1. 输入

用户从页面输入的用户信息，包括用户名和密码，其中用户名可以是注册时的用户名或者邮箱，如果是用户名，则需要满足字符长度至少为6、至多为20，密码需要满足字符长度至少为6、至多为30。

1. 处理

用户填写的表单数据由前端传递给后端；

IF ( 输入的用户名/邮箱和密码在账户数据库中不匹配 )

{

认证失败，在页面显示提示信息“输入的用户名与密码不匹配”；

}

将认证(登录)成功的信息返回给前端。

1. 输出

当输入的用户名/邮箱与密码不匹配时，在页面上显示“输入的用户名与密码不匹配”的提示信息；当输入的用户名/邮箱与密码匹配时，登录成功，并将认证成功的信息返回给前端，即页面跳转到用户主页。

### 3.1.3 账户管理：用户登出

1. 介绍

用户在使用完毕管理功能后，可登出账户，避免他人误操作用户数据。

1. 输入

用户在页面上点击“退出”按钮。

1. 处理

将当前用户的登录状态注销。

1. 输出

页面跳转到登录页面。

### 3.1.4 项目管理：新建项目

1. 介绍

一个用户可以新建多个项目，每个项目下可以管理多个模型。首先要有项目，才可以添加管理部署模型，所以新建项目往往是新用户登录后最先做的事；另外不同的模型可能隶属不同的项目，也需要不断新建项目。

1. 输入

选择“新建项目”后，输入项目名称和项目描述，均为字符信息，其中项目描述非必须输入。

1. 处理

从前端获取数据传递至后端，检查项目名称有效性，如果该用户不存在同名的项目，则添加新的用户-项目信息，否则返回失败信息。

1. 输出

若添加成功则输出成功信息，否则输出具体的错误信息。

### 3.1.5 项目管理：查看项目

1. 介绍

用户登录后，可以在自己的项目空间下看到自己所有的项目信息，包括项目名称和简介。在查看项目后，可以进一步查看具体某个项目下的模型信息。

1. 输入

用户登录状态下用户名，选择“项目列表”。

1. 处理

根据用户名称获取用户名下所有项目的项目名和描述信息。

1. 输出

用户名下没有任何项目时显示为空，否则显示刷新后的项目信息。

### 3.1.6 项目管理：删除项目

1. 介绍

本需求实现用户项目管理中删除项目的需求。当该项目及项目下的模型和实例都不再需要时，用户可以删除项目。

1. 输入

选择要删除的用户名下的某一项目（主要信息是项目名）。

1. 处理

后端获取要删除的项目后，在用户所有的项目中删除该项目的信息，包括项目基本信息，以及项目下包括的模型和实例的信息。

1. 输出

输出删除成功与否的信息。

### 3.1.7 项目管理：更新项目

1. 介绍

用户在新建项目一段时间后，可能发现当初新建项目时填写的信息并不准确，而删除项目重建会影响到项目下包含的模型和实例，所以有修改更新项目的需求。

1. 输入

用户要修改的项目的具体的部分（如项目名、详细信息）。

1. 处理

将用户名下所有涉及到相关信息的位置全部更新，比如修改项目名，则关联的模型和实例等对应的项目名也要修改。

1. 输出

输出更新成功与否的信息。

### 3.1.8 模型管理：在项目中导入模型

1. 介绍

本功能要求能够在指定项目中导入模型文件，支持导入多个模型。规定同一个模型只能导入一次，通过模型名称和版本进行判断。

1. 输入

用户输入：

a) 模型名称：一个模型仅有一个名称；

b) 模型描述：用户对模型的备注，可以为空；

c) 模型类型：以选择的形式填入模型的类别，以便系统识别；

d) 模型文件：需要上传部署的模型文件；

e) 模型版本：在模型名称相同时，模型版本自动+1.

系统获取：

a) 项目编号；

b) 用户状态。

1. 处理
2. 输入数据的有效性检测：

检测对应项目编号是否为该用户能够合法获取的项目，防止非法入侵；

检测输入模型类型是否为系统指定类型范围；

检测模型文件是否合法；

检测用户是否在合法登录状态。

1. 对异常情况的回应：

类型错误：“请重新选择类型”；

文件非法：“文件格式错误，请重新上传文件”；

用户登录状态异常：“用户未登录”；

溢出、通讯失败等服务器错误：“服务器错误，请稍候”。

1. 操作序列

①将模型信息set到数据库中；

②将模型文件保存至服务器，并生成文件路径；

③将文件路径保存到数据库。

1. 对文件是否成功保存至数据库进行检查：

IF ( 能够根据文件路径找到文件 )

返回模型导入成功

ELSE IF ( 只能找到文件找不到模型信息 )

再次set模型信息到数据库

ELSE IF ( 只能找到模型信息找不到模型文件 )

再次将文件保存，生成文件路径，保存到数据库

ELSE 再次执行set模型信息和保存文件两步

1. 输出

确认模型信息存入数据库后，返回模型编号和“导入成功”信息，否则根据对应错误返回错误信息。

### 3.1.9 模型管理：对模型部署的参数进行设置

1. 介绍

要求可以对模型部署的一些参数进行配置，以便于后续部署

1. 输入

用户输入：

a) 模型版本：选择之后要部署的模型的版本；

b) 网络服务运行环境：选择模型运行的网络环境；

c) 预留CPU：是否预留CPU，是，再填入预留CPU的个数；

d) 预留内存：是否预留内存，是，再填入预留内存的大小。

系统获取：

a) 模型编号：对应版本的模型编号；

b) 用户状态；

c) 项目编号。

1. 处理
2. 输入数据的有效性检测：

检测对应项目编号和模型编号是否为该用户能够合法获取的项目或模型，防止非法入侵；

检测输入模型版本是否存在；

检测预留CPU个数是否在允许范围内；

检测预留内存大小是否允许；

检测用户是否在合法登录状态。

1. 对异常情况的回应：

版本不存在：“版本信息错误，请重新选择”；

CPU个数不足：“CPU余量不足，请重新选择”；

内存不足：“内存大小不足，请重新选择”；

用户登录状态异常：“用户未登录”；

溢出、通讯失败等服务器错误：“服务器错误，请稍后”。

1. 对文件是否成功保存至数据库进行检查：

IF ( 查询到数据库中保存了相关配置 )

返回设置成功

ELSE 重新将信息设置到相关模型的数据库中

1. 输出

确认模型参数设置存入数据库后，返回模型编号和“设置成功”信息，否则根据对应错误返回错误信息。

### 3.1.10 模型管理：查看某一项目下所有模型及相关配置

1. 介绍

用户可以查看某一项目下所有的模型文件，并查看模型相关的配置信息、参数信息、版本号等。

1. 输入

用户输入：

a) 项目编号：选择的项目的对应编号；

b) 模型编号：选择的模型的对应编号。

系统获取：

用户状态；

1. 处理

a) 输入数据的有效性检测：

检测对应项目编号和模型编号是否为该用户能够合法获取的项目或模型，防止非法入侵；

检测用户是否合法登录。

b) 对异常情况的回应：

用户登录状态异常：“用户未登录”；

溢出、通讯失败等服务器错误：“服务器错误，请稍后”。

c) 从数据库查找对应项目、模型信息，并按格式返回

1. 输出

按照对应格式返回项目信息、模型信息，否则根据对应错误返回错误信息。

### 3.1.11 模型管理：删除模型

1. 介绍

用户能够删除项目中指定模型

1. 输入

用户输入：模型编号；

系统获取：用户状态。

1. 处理
2. 输入数据的有效性检测：

检测对应项目编号和模型编号是否为该用户能够合法获取的项目或模型，防止非法入侵；

检测用户是否合法登录。

1. 对异常情况的回应：

用户登录状态异常：“用户未登录”；

溢出、通讯失败等服务器错误：“服务器错误，请稍后”。

1. 操作序列：

①根据模型编号找到对应模型文件；

②检查模型是否处于部署服务状态，如果是，关掉服务，并释放对应资源；

③删除模型文件；

④将数据库中关于该模型的配置、部署记录等信息标为不可访问。

1. 检测模型是否成功删除：

IF ( 模型文件存在 )

删除文件

ELSE IF ( 数据库中相关信息仍可访问 )

再次设置为不可访问

ELSE 返回“模型删除成功”

1. 输出

确定删除模型数据后，返回模型编号和“模型成功删除”，否则根据对应错误返回错误信息。

### 3.1.12 实例管理：启动实例

1. 介绍

本需求实现从静态模型文件到动态实例的启动过程，包括对相应配置文件的加载和应用。模型静态文件是指用户通过门户前端上传到服务器的若干文件，包括静态模型结构以及参数变量文件。门户系统会储存对应模型在服务器上的位置，启动过程即在索引到文件位置后使用对应的机器学习运行时加载改模型并使其常驻端口，新建后台运行的Flask程序用于响应用户发出的模型使用请求，此时成为一个具体的实例。上述过程包括配置文件的读取以及个性化加载过程。期间需要注意如下异常：（a）模型文件地址错误；（b）配置项错误；（c）模型加载失败；（d）后台Flask程序生成失败；（e）端口被占用。抛出异常后向门户后台返回响应的错误码。

2. 输入

从门户后端传入的项目编号。触发条件为用户在某一项目下点击“部署”按钮。

3. 处理

根据项目编号索引模型文件夹

根据项目编号索引配置文件地址

Try{

读取模型文件夹

}

Exception 文件夹地址错误{返回错误码 4031}

IF（配置文件地址不为空）{

Try{

读取配置文件

} exception 文件地址错误{返回错误码 4032}

Try{

对象化配置

} exception 配置项错误{返回错误码 4033}

}

选择相应的机器学习平台运行时

Try{

初始化一个新的Flask程序

记录该Flask程序的pid，即进程编号

}exception 初始化错误{返回错误码 4034}

基于该Flask程序加载模型

Try{

在给定范围内随机生成端口号

}exception 无多余端口号{返回错误码 4035}

更改Flask程序的路由配置

返回门户后端处理结果及生成的服务URL

4. 输出

部署系统一定会给门户后端返回处理结果的代码，在启动过程成功的情况下还会包含请求服务的URL地址。

### 3.1.13 实例管理：删除实例

1. 介绍

本需求实现对已启动或处于暂停状态的实例进行删除的操作。删除操作后，实例不再消耗服务器内存及计算资源，也不再响应用户的服务请求。可能出现的异常包括（a）不存在该实例；（b）该实例正在使用中，无法删除；（c）实例进程删除失败。

2. 输入

实例编号，触发条件为用户在某一实例下点击“删除”按钮。

3. 处理

Try{

根据实例编号索引实例

}

Exception 不存在该实例{返回错误码 4041}

IF ( 该实例正在被用户请求服务 ){

返回错误码 4042

}

读取该实例运行的进程编号

Try{

杀掉该进程

删除对应实例记录

}

Exception 删除实例失败{返回错误码 4043}

返回门户后端处理结果代码 4044

4. 输出

返回门户后端响应的错误码。删除成功时返回处理结果代码4044.

### 3.1.14 实例管理：暂停运行中实例

1. 介绍

本需求实现对运行中的模型部署实例进行暂停服务的功能。对上传成功的模型进行一次部署即可得到一个运行中的实例，对外提供服务。暂停运行中的实例是指停止该实例对外提供的服务，使外界无法向该实例发送请求并获得响应。

2. 输入

在前端点击“暂停实例”按钮，发起暂停实例的请求，其中包含了该实例的ID值。

3. 处理

收到暂停实例请求；

IF ( 实例ID不是正整数 or 数据库中不存在该实例ID or 该实例已处于暂停状态 ) {

返回暂停失败的响应；

}

屏蔽该实例所监听的端口；

更新数据库中该实例的状态；

返回暂停成功的响应；

4. 输出

请求暂停的实例ID合法且该实例处于运行中时，屏蔽该实例所监听的端口，暂停实例服务；否则返回暂停失败的响应。

### 3.1.15 实例管理：恢复暂停中实例

1. 介绍

本功能实现对暂停中的实例进行恢复服务的功能。恢复暂停中的实例是指恢复实例对外界提供的服务，使实例可以响应外界发送过来的请求。

2. 输入

在前端点击“恢复实例”按钮，发起恢复实例的请求，其中包含了该实例的ID值。

3. 处理

收到恢复实例请求；

IF (实例ID不是正整数 or 数据库中不存在该实例ID or 该实例已处于运行中状态) {

返回恢复失败的响应；

}

打开该实例所监听的端口；

更新数据库中该实例的状态；

返回恢复成功的响应；

4. 输出

请求恢复的实例ID合法且该实例处于暂停中时，打开该实例所监听的端口，恢复实例服务；否则返回恢复失败的响应。

### 3.1.16 使用已部署的模型

1. 介绍

用户可通过REST API向服务器发送已部署模型的使用请求。

2. 输入

待识别的图像。

3. 处理

将待识别的图像传入正在运行的深度学习模型实例，获得模型运行后的识别结果。若模型实例不存在或已暂停，则向用户返回错误消息。

4. 输出

若用户所请求的深度学习模型正在运行，则输出图片的识别结果，否则输出错误消息。

## 3.2 性能需求

### 3.2.1 同时运行模型实例数

本软件最大支持同时运行100个深度学习模型。

### 3.2.2 网站界面并发访问能力

软件对外展现的网站界面最大支持每分钟500次的访问频率。

### 3.2.3 REST API并发访问能力

通过REST API向运行中的模型实例发送请求的频率最高为每分钟100次。

## 3.3 外部接口需求

### 3.3.1 用户接口

1. 浏览器前端页面

前端页面提供了以下操作：

账户管理：用户通过点击相应按钮和填写表单，执行注册、登录、登出账户；

项目管理：用户通过点击相应按钮和填写表单，执行新建、查看、删除项目；

模型管理：用户通过点击相应按钮和填写表单，执行a) 在项目中导入模型，b) 对模型部署的参数进行设置，c) 将模型部署到服务器，d) 查看某一项目下所有模型及相关配置，e) 暂停已部署模型的服务，f) 删除模型；

实例管理：用户通过点击相应按钮执行实例的启动和删除。

1. REST API

用户通过HTTP的POST请求向REST API发送待检测图像，并通过此API获得神经网络的判断结果。

### 3.3.2 软件接口

1. 数据库与门户系统后端的接口

软件名称：MySQL数据库

版本号：5.7

用途：用于与门户系统后端的数据管理，实现数据的一致性和可维护性。

1. 查询数据接口

门户系统后端在需要查询数据的时候用到本接口，可定义查询表、过滤条件、限制条件、排序条件等。接口将‘SQLAlchemy’转成‘SQL’语句，带入MySQL数据库中进行查询。

**表1 查询数据接口**

|  |  |
| --- | --- |
| 接口名称 | 接口作用 |
| session.query(表名).filter(过滤条件).all() | 全部符合条件的数据 |
| session.query(表名).filter(过滤条件).first() | 第一个符合条件的数据 |
| session.query(表名).filter(过滤条件).one() | 结果为一个时正常，多了就报错 |
| session.query(表名).filter(过滤条件).slice(n,m).all() | 查到的数据做切片返回 |
| session.query(表名).filter(过滤条件).order\_by(列名).all() | 根据某一列排序返回数据 |

1. 修改数据接口

门户系统后端在需要修改数据的时候用到本接口。接口将‘SQLAlchemy’转成‘SQL’语句，带入MySQL数据库中进行查询。

**表2 修改数据接口**

|  |  |
| --- | --- |
| 接口名称 | 接口作用 |
| session.query(表名).filter(过滤条件).update({"列名":"修改数据"}) | 修改查询到的数据 |
| session.commit() | 提交修改 |

1. 添加数据接口

门户系统后端在需要添加一行数据的时候用到本接口。接口将‘SQLAlchemy’转成‘SQL’语句，带入MySQL数据库中进行查询。

**表3 添加数据接口**

|  |  |
| --- | --- |
| 接口名称 | 接口作用 |
| session.add’(表名’(列名1=’’,列名2=’’…)) | 修改查询到的数据 |
| session.commit() | 提交修改 |

1. 删除数据接口

门户系统后端在需要删除指定数据的时候用到本接口。接口将‘SQLAlchemy’转成‘SQL’语句，带入MySQL数据库中进行查询。

**表4 删除数据接口**

|  |  |
| --- | --- |
| 接口名称 | 接口作用 |
| session.delete(对象) | 删除一行数据 |
| session.commit() | 提交修改 |

### 3.3.3 硬件接口

本软件属于上层服务软件，不对外提供访问硬件的接口。

### 3.3.4 通讯接口

本软件不对外提供通信接口。

# 4 总体设计约束

## 4.1 标准符合性

1. 网站前端的数据传输遵循RFC 2616 – HTTP/1.1协议规范；
2. 访问模型部署功能的REST API遵循RFC 3986通用URI语法；

## 4.2 硬件约束

运行本软件所需的最小硬件资源：

Intel Core i3 8100或同等级的中央处理器；

NVIDIA GeForce GTX 760或同等级的图形处理器；

2 GB 随机存储器；

40 GB可用硬盘空间。

## 4.3 软件约束

运行本软件系统所需的软件资源：

Ubuntu 18.04；

Docker 1.23.2；

Kubernetes 1.13.5.

# 软件质量特征

## 5.1兼容性

作为一个模型自动化部署工具，客户的服务器类型，软件环境多种多样，本系统需要兼容不同的部署环境。系统需要将自身的依赖项以及代码打包完整，采用容器技术部署在用户的服务器资源上，这样不会影响用户服务器本身的环境和在运行的程序。

## 5.2 可扩展性

随着用户部署的模型服务数量增多，用户可能需要新增服务器资源。系统需要支持这种可插拔的扩展方式，同一调度服务器资源，均衡部署模型服务。

## 5.3 容灾性

系统需要对某一个服务器的宕机所导致的服务不可用做出响应。具体来说，某一服务器节点的宕机会导致部署在其上的模型服务失效，为了确保用户部署的模型服务的可用性，系统需要监控每个服务器节点的状态，并在节点失效时对其上的模型服务进行重新部署，以保证模型服务的可用性。

# 附录 A Flask框架分析

## A.1 框架介绍

Flask是当下流行的Web框架，它是用Python实现的。Flask最显著的特点包括它是一个“微”框架(microframework)。“微”意味着Flask旨在保持核心功能的简单，因此它又是一个易于扩展的框架。

在“微”框架的情况下，Flask不包含数据库抽象层、表单验证，或者是其它任何已有多种库可以胜任的功能。然而，Flask支持用扩展来给本项目添加这些功能，无需局限于特定的框架功能，让该项目更具有个性化的特征。众多Flask扩展包提供了数据库集成、表单验证、上传处理以及各种各样的开放认证技术等功能，甚至可以根据项目需求自行开发一些基于Flask框架的功能包，具有极强的拓展性和定制性，同时Flask可以很好地结合MVC模式进行开发，开发人员分工合作，小型的团队在短时间内就可以完成功能丰富的中小型网站或Web服务的实现。这些特性，使得它在Web开发方面变得非常流行，也成为本项目的首选框架。

## A.2 框架核心

Flask框架的核心包括两个部分，Werkzeug和Jinja2，Werkzeug是一个WSGI(Python Web Server Gateway Interface)的工具包，它可以作为一个Web框架的底层库，其中封装了很多Web服务的工具，包括最重要的会话模块的Response、Resquest等。

Jinja2是Python语言的一个通用模版引擎，具有一系列强大的功能的同时还具有灵活性、快速性、安全性的优点，因此被广泛使用。其灵活性主要体现在Jinja2提供了控制结构、表达式和继承等，快速性在于其仅有控制结构，不允许在模版中编写太多业务逻辑，可读性很强。

## A.3 Flask通用扩展包

Flask作为一个微框架，具有极强的可扩展性，因此下面介绍本项目中会用到的Flask的扩展插件。

1. Flask-SQLAlchemy数据库扩展包

数据库是长期存储在计算机内，大量有组织可共享的数据的集合。SQLAlchemy是一个关系型数据库框架，它提供了高层的ORM(Object Relational Mapping)和底层的原生数据库的操作， 实际上是对数据库的抽象，让开发者不用直接和 SQL 语句打交道，而是通过 Python对象来操作数据库，在舍弃一些性能开销的同时，换来的是开发效率的较大提升。而Flask-SQLAlchemy则是一个基于SQLAlchemy操作的Flask扩展包。

1. Flask-WTF表单扩展包

Flask\_WTF是Flask框架的表单验证模块，可以很方便地生成表单，也可以当做json数据交互的验证工具，支持热插拔。Flask-WTF其实是对wtforms组件的封装，而wtfroms其实是一个支持多种web框架的form组件，主要用于对用户请求数据的进行验证，依照功能类别来说wtforms分别由表5所示的几个类别构成。

**表5 wtforms的构成**

|  |  |
| --- | --- |
| Forms | 主要用于表单验证、字段定义、HTML生成，并把各种验证流程聚集在一起进行验证 |
| Fields | 主要负责渲染(生成HTML)和数据转换 |
| Validator | 主要用于验证用户输入的数据的合法性。比如Length验证器可以用于验证输入数据的长度 |
| Widgets | html插件，允许使用者在字段中通过该字典自定义html小部件 |
| Meta | 用于使用者自定义wtforms功能，例如csrf功能开启 |
| Extensions | 丰富的扩展库，可以与其他框架结合使用 |

# 参考文献

[1] 中国国家标准化委员会.计算机软件需求规格说明：GB/T 9385-2008 [S].

[2] 中国国家标准化委员会.计算机软件文档编制规范：GB/T 8567-2006 [S].

[3] 中国人民解放军总装备部.军用软件开发文档通用要求：GJB 438B-2009 [S].

[4] Flask documentation [EB/OL]. https://flask.palletsprojects.com/en/1.1.x/

[5] Representational State Transfer [EB/OL]. http://en.wikipedia.org/wiki/Representational\_state\_transfer