**编 号：**

**审定成绩：**

|  |  |  |  |
| --- | --- | --- | --- |
| **重庆邮电大学**  **毕业设计（论文）** | | | |
|  | | | |
|  | **中文题目** | **[基于AI语音识别的websocket聊天室](http://172.20.2.53:84/StudentSelectTopic/javascript:void(0))** |  |
|  |  |
| **英文题目** |  |
|  |  |
| **学院名称** | **软件工程学院** |
| **学生姓名** | **陈俊松** |
| **专 业** | **英语+软件** |
| **班 级** | **1301415** |
| **学 号** | **2014213963** |
| **指导教师** | **宋琦** |
| **答 辩 组**  **负 责 人** |  |
|  | |

**年 月**

**重庆邮电大学教务处制**

软件工程学院本科毕业设计(论文)诚信承诺书

本人郑重承诺：

我向学院呈交的论文《 基于AI语音识别的websocket聊天室 》，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明并致谢。本人完全意识到本声明的法律结果由本人承担。

年级 2014级

专业 英语+软件

班级 1301415

承诺人签名

年 月 日

学位论文版权使用授权书

本人完全了解重庆邮电大学有权保留、使用学位论文纸质版和电子版的规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。本人授权重庆邮电大学可以公布本学位论文的全部或部分内容，可编入有关数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编本学位论文。

（注：保密的学位论文在解密后适用本授权书。）

|  |  |
| --- | --- |
| 学生签名： | 指导老师签名： |
| 日期： 年 月 日 | 日期： 年 月 日 |

摘要

随着时代的飞速发展，互联网技术日新月异，传统的通信技术已经不能满足我们在日常通信的要求，WebSocket 即时通信技术应运而生，而基于语音识别的即时通信技术更是将我们的生活丰富化，便捷化。技术的发展，引导我们不断的进步。WebSocket 协议本质上是一个基于 TCP 的协议。为了建立一个 WebSocket连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP请求不同，包含了一些附加头信息，其中附加头信息“Upgrade: WebSocket”表明这是一个申请协议升级的 HTTP请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket便可建立连接。双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。语音识别技术，也被称为自动语音识别Automatic Speech Recognition，(ASR)，就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令的高技术。　主要包括特征提取技术、模式匹配准则及模型训练技术三个方面。其目标是将人类的语音中的词汇内容转换为计算机可读的输入。

基于AI语音识别的Websocket聊天室主要是通过语音识别技术将我们的声音进行语音识别并及时进行数据传输到服务端，服务端通过验证进行语音识别，并返回响应，还可以通过机器人智能回复，智能查询相关信息，适用于多个应用场景，可流畅的在各个系统上运行。

基于AI语音识别的Websocket聊天室主要是基于C/S架构，前端通过react + ant.design 进行MVVM模式构建，后端通过express + websocket进行服务端实现，数据库通过NO-SQL的MongoDB的进行搭建。

**关键词：**Websocket，语音识别，聊天室

**Abstract**

With the rapid development of the times, Internet technology is changing with each passing day, traditional communication technologies can no longer meet the requirements of our daily communications, WebSocket instant messaging technology came into being, and instant messaging technology based on speech recognition enriches our lives. Facilitation. The development of technology leads us to continuous improvement. The WebSocket protocol is essentially a TCP-based protocol. In order to establish a WebSocket connection, the client browser first sends an HTTP request to the server. This request is different from the usual HTTP request and contains some additional header information. The additional header “Upgrade: WebSocket” indicates that this is an application protocol. Upgraded HTTP requests. The server parses these additional headers and generates response information back to the client. WebSockets on the client and server can establish connections. Both parties can freely transmit information through this connection channel, and this connection will continue to exist until either the client or the server side actively closes the connection. Speech recognition technology, also known as Automatic Speech Recognition (ASR), is a high technology that allows machines to convert speech signals into corresponding text or commands by identifying and understanding the process. It mainly includes three aspects: feature extraction technology, pattern matching criteria, and model training technology. The goal is to convert the vocabulary content of human speech into computer-readable input.

The Websocket chat room based on AI voice recognition mainly uses voice recognition technology to carry out voice recognition of our voices and transmit data to the server in time. The server authenticates and recognizes voices and returns responses. It can also reply intelligently through robots. Query related information, suitable for multiple application scenarios, can run smoothly on various systems.

The Websocket chat room based on AI voice recognition is mainly based on the C/S architecture. The front end constructs the MVVM mode through react + ant.design, and the back end implements the server through express + websocket. The database is constructed through NO-SQL MongoDB

**Keywords**: Websocket , voice recognition, chat room

目录

[第1章 引言 1](#_Toc5562)

[1.1 研究背景和意义 1](#_Toc18605)

[1.2 国内外研究现状 3](#_Toc14563)

[1.2.1 国外研究现状 3](#_Toc12678)

[1.2.2国内研究现状 4](#_Toc32381)

[1.3 主要内容和工作安排 5](#_Toc31460)

[第2章开发工具及平台兼容性 6](#_Toc1017)

[2.1编码工具和调试工具的选择 6](#_Toc13642)

[2.2.1 sublime的开发优势 6](#_Toc9563)

[2.2.1 基于chrome扩展程序React Developer Tools的调试 6](#_Toc19467)

[2.2 Node.js和npm介绍 8](#_Toc10880)

[2.2.1 Node.js的介绍和开发优势 8](#_Toc12806)

[2.2.2 npm包管理工具的使用介绍 9](#_Toc9078)

[2.3打包工具webpack介绍 10](#_Toc9486)

[2.4代码规范及ESLint介绍使用 10](#_Toc5310)

[2.5 MongoDB数据库介绍 11](#_Toc17844)

[第3章系统分析与设计 12](#_Toc23152)

[3.1 系统需求分析 12](#_Toc28244)

[3.2系统概要设计 12](#_Toc8592)

[3.2.1 系统结构设计 12](#_Toc19345)

[第4章 系统的具体实现 14](#_Toc4386)

[4.1 前端的具体实现 14](#_Toc2466)

[4.1.1 音频转换模块 14](#_Toc18617)

[4.1.2 语音识别模块 18](#_Toc17697)

[4.1.3 语音朗读模块 19](#_Toc18577)

[4.1.4 登录注册模块 21](#_Toc14202)

[4.1.5 聊天室模块 23](#_Toc21619)

[4.2 后端express + websocket设计 24](#_Toc23837)

[4.1.1 websocket设计模块 24](#_Toc7757)

[4.1.2 语音识别接口模块 24](#_Toc7011)

[4.1.3 机器人回复模块 25](#_Toc11174)

[4.1.4 登录注册模块 26](#_Toc18428)

[4.1.5 聊天室模块 27](#_Toc8328)

[4.1.6 数据库连接模块 27](#_Toc3944)

[第5章 系统测试 28](#_Toc5548)

[5.1软件测试的目的 29](#_Toc5351)

[5.2软件测试的方法 29](#_Toc272)

[5.3测试方案 30](#_Toc24375)

[5.4测试用例 31](#_Toc31822)

[5.5 测试结论 32](#_Toc24595)

[总结 32](#_Toc7829)

[参考文献 33](#_Toc5321)

[致谢 34](#_Toc3952)

# 

# 第1章 引言

## 1.1 研究背景和意义

大概在 1992 年，一家称作 Nombas 的公司开发了一种叫做 C 减减（C-minus-minus，简称 Cmm）的嵌入式脚本语言。Cmm 背后的理念很简单：一个足够强大可以替代宏操作（macro）的脚本语言，同时保持与 C （和 C ++）足够的相似性，以便开发人员能很快学会。这个脚本语言捆绑在一个叫做 CEnvi 的共享软件中，它首次向开发人员展示了这种语言的威力。Nombas 最终把 Cmm 的名字改成了 ScriptEase。当 Netscape Navigator 崭露头角时，Nombas 开发了一个可以嵌入网页中的 CEnvi 的版本。这些早期的试验被称为 Espresso Page（浓咖啡般的页面），它们代表了第一个在万维网上使用的客户端语言。而 Nombas 丝毫没有料到它的理念将会成为万维网的一块重要基石。

当网上冲浪越来越流行时，对于开发客户端脚本的需求也逐渐增大。此时，大部分因特网用户还仅仅通过 28.8 kbit/s 的调制解调器连接到网络，即便这时网页已经不断地变得更大和更复杂。而更加加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器进行多次地往返交互。设想一下，用户填完一个表单，点击提交按钮，等待了 30 秒的处理后，看到的却是一条告诉你忘记填写一个必要的字段。那时正处于技术革新最前沿的 Netscape，开始认真考虑开发一种客户端脚本语言来解决简单的处理问题。

就在 Netscape Navigator 2.0 即将正式发布前，Netscape 将其更名为 JavaScript，目的是为了利用 Java 这个因特网时髦词汇。Netscape 的赌注最终得到回报，JavaScript 从此变成了因特网的必备组件。

随着javascript的迅速发展，其在互联网的角色也越来越鲜明，重要性也越来越高，由此在互联网领域的作用日渐突出。根据许多调查显示，JavaScript连续多年在互联网开发中高居榜首。

‘**能用 JavaScript 编写的程序，都终将会以 JavaScript 编写**’，这句话曾经听起来十分高傲，但是随著 Node.js 的发展，我们渐渐地明白到这句话的远见性──JavaScript 真的越来越强大了。

2009年2月，Ryan Dahl在博客上宣布准备基于V8创建一个轻量级的[Web服务器](https://www.baidu.com/s?wd=Web%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBmW6znywWPy7-nhNWPWm30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHDLP1fLPHbdP1mYPW01rHcdrf" \t "https://zhidao.baidu.com/question/_blank)并提供一套库。 2009年5月，Ryan Dahl在GitHub上发布了最初版本的部分Node.js包，随后几个月里，有人开始使用[Node.js开发](https://www.baidu.com/s?wd=Node.js%E5%BC%80%E5%8F%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBmW6znywWPy7-nhNWPWm30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHDLP1fLPHbdP1mYPW01rHcdrf" \t "https://zhidao.baidu.com/question/_blank)应用。 2009年11月和2010年4月，两届JSConf大会都安排了Node.js的讲座。 2010年年底，Node.js获得[云计算](https://www.baidu.com/s?wd=%E4%BA%91%E8%AE%A1%E7%AE%97&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBmW6znywWPy7-nhNWPWm30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHDLP1fLPHbdP1mYPW01rHcdrf" \t "https://zhidao.baidu.com/question/_blank)服务商Joyent资助，创始人Ryan Dahl加入Joyent全职负责Node.js的发展。 2011年7月，Node.js在微软的支持下发布Windows版本。Node.js 是一个非常新兴的开发工具，年龄远不如 Python、Ruby、PHP 等老大哥，但是它确实有史以来发展最快的开发工具，没有之一。在这短短的几年间，我们看到了 Node.js 从当初的一无所有到如今的飞速发展，这是没有任何其他开发工具能够媲美的。

HTML5草案的前身名为 Web Applications 1.0，于2004年被WHATWG提出，于2007年被W3C接纳，并成立了新的 HTML 工作团队。HTML 5 的第一份正式草案已于2008年1月22日公布。HTML5 仍处于完善之中。然而，大部分[现代浏览器](https://baike.baidu.com/item/%E7%8E%B0%E4%BB%A3%E6%B5%8F%E8%A7%88%E5%99%A8" \t "https://baike.baidu.com/item/html5/_blank)已经还仍不具备 HTML5 支持。

“语音识别”的终极梦想，是真正能够理解人类语言甚至是方言环境的系统。但几十年来，人们并没有一个有效的策略来创建这样一个系统，直到人工智能技术的爆发。在过去几年中，人们在人工智能和深度学习领域的突破，让语音识别的探索跨了一大步。其中的精髓便是对话式AI，对话式AI包含有两个主要类别：人机界面，以及人与人沟通的界面。在人机界面中，人类与机器往往通过语音或文本交互，届时机器会理解人类 (尽管这种理解方式是有限的) 并采取相应的一些措施。

基于HTML5的版本兼容性和JavaScript以及AI语音识别技术的不断演变，为了实现真正的纯JavaScript语言演绎的PC端语音识别系统，由此开发了基于AI语音识别的聊天室。

基于AI语音识别的聊天室摒弃了传统的JSP开发模式，前端页面和后端服务器接口都采用纯JavaScript语言编写，实现技术的一致性，减轻前后端数据连通的复杂性，可以使开发人员清晰的了解数据的流动，实现前后端的无缝式开发。而前端部分通过对HTML5标准进行二次包装，以实现语音格式转换，语音朗读TTS引擎的兼容性，语音识别部分通过封装的AI接口，智能识别并返回机器人响应。完善了HTML5标准的部分接口和AI语音识别的信息响应，实现了纯PC端JavaScript开发的AI语音识别聊天室。

## 1.2 国内外研究现状

### 1.2.1 国外研究现状

语音识别的研究工作可以追溯到20世纪50年代AT&T贝尔实验室的Audry系统，它是第一个可以识别十个英文数字的语音识别系统。但真正取得实质性进展，并将其作为一个重要的课题开展研究则是在60年代末70年代初。实验室语音识别研究的巨大突破产生于20世纪80年代末：人们终于在实验室突破了大词汇量、连续语音和非特定人这三大障碍，第一次把这三个特性都集成在一个系统中，比较典型的是卡耐基梅隆大学（CarnegieMellonUniversity）的Sphinx系统，它是第一个高性能的非特定人、大词汇量连续语音识别系统。

20世纪90年代前期，许多著名的大公司如IBM、苹果、AT&T和NTT都对语音识别系统的实用化研究投以巨资。语音识别技术有一个很好的评估机制，那就是识别的准确率，而这项指标在20世纪90年代中后期实验室研究中得到了不断的提高。比较有代表性的系统有：IBM公司推出的ViaVoice和DragonSystem公司的NaturallySpeaking，Nuance公司的NuanceVoicePlatform语音平台，Microsoft的Whisper，Sun的VoiceTone等。

其中IBM公司于1997年开发出汉语ViaVoice语音识别系统，次年又开发出可以识别上海话、广东话和四川话等地方口音的语音识别系统ViaVoice‘98。它带有一个32，000词的基本词汇表，可以扩展到65，000词，还包括办公常用词条，具有“纠错机制”，其平均识别率可以达到95%。该系统对新闻语音识别具有较高的精度，是目前具有代表性的汉语连续语音识别系统。

2002 年 8 月，W3C 建立了第 4 个 HTML 工作组 （实际上是第 3 个 XHTML 工作组），目标是完成 HTML 到 XHTML 的转变，包括完成以尽可能多的 使用通用的 XML 技术为设计目标的 XHTML 2.0。同 时，HTML 工作组发布了 XHTML 2.0 第一个公开工 作草案，并在声明中指出，尽管 XHTML 2.0 来源于 HTML 4、XHTML 1.0 和 XHTML 1.1，XHTML 2.0 并 不向后兼容早期版本。

HTML 5 在 HTML 4 的基础上新增了较多元素和 属性，同时，也废除了部分元素和属性。与 HTML 4 相比，HTML 5 在文档结构、页面元素、图形支持、 多媒体支持、本地存储、离线应用、多线程和地理位置信息等方面有较大的变化，这些变化极大地提 升了 HTML 5 对 Web 应用的开发支持。

浏览器是 HTML 5 的重要解析器，因此，浏览 器对 HTML 5 的支持直接影响 HTML 5 的应用。由 于主要浏览器厂商直接参与了 HTML 5 的标准制定 工作，因此，多数浏览器厂商发布的新版本浏览器 都加强了对 HTML 5 的支持。

### 1.2.2国内研究现状

我国语音识别研究工作起步于五十年代，但近年来发展很快。研究水平也从实验室逐步走向实用。从1987年开始执行国家863计划后，国家863智能计算机专家组为语音识别技术研究专门立项，每两年滚动一次。我国语音识别技术的研究水平已经基本上与国外同步，在汉语语音识别技术上还有自己的特点与优势，并达到国际先进水平。

中科院自动化所、声学所、清华大学、北京大学、哈尔滨工业大学、上海交通大学、中国科技大学、北京邮电大学、华中科技大学等科研机构都有实验室进行过语音识别方面的研究，其中具有代表性的研究单位为清华大学电子工程系与中科院自动化研究所模式识别国家重点实验室。

HTML 5 标准涉及浏览器开发商、工具开发商及 应用开发商等，技术覆盖面较广，标准制定难度较大。 我国为了推动基于 HTML 5 的互联网和移动互联网 产业发展，我国标准研制主要聚焦于聚焦于浏览器兼 容性规范、浏览器编程接口及标准符合性测试规范 等基础性标准，并结合 HTML 5 的后续发展，逐步 建立起符合我国产业发展需求的标准体系。

目前，HTML 5 标准由 W3C 和 WHATWG 联合 制定。我国的百度、腾讯、UC、华为、盛大、360 等互联网企业已于近年加入 W3C，并派代表参加 HTML 工作组。

## 1.3 主要内容和工作安排

本文主要按以下方式组织：

第一章：主要介绍系统的研究目的和意义，研究背景，研究现状，以及标准实现的不同情况。

第二章：主要介绍系统实现的所需工具优势，调试软件的使用，包管理器的开发和使用，打包工具的使用，以及数据库开发的优势

第三章：主要介绍系统的设计方向，包括系统需求分析设计，系统概要设计，系统结构设计，数据库设计方向。

第四章：主要介绍系统的编码实现，包括HTML5标准的实现以及兼容处理，包括对不支持浏览器的容错处理，语音识别的格式转换问题，前端框架架构处理，后端接口数据处理，以及机器人回复处理。

第五章：主要介绍系统测试结果，包括语音识别结果，服务器压力测试，系统黑盒和白盒测试，数据库相关测试。

# 第2章开发工具及平台兼容性

## 2.1编码工具和调试工具的选择

## 2.2.1 sublime的开发优势

Sublime Text 是一个代码编辑器，也是HTML和散文先进的文本编辑器。Sublime Text是由程序员Jon Skinner于2008年1月份所开发出来，它最初被设计为一个具有丰富扩展功能的[Vim](https://baike.baidu.com/item/Vim" \t "_blank)。

Sublime Text具有漂亮的用户界面和强大的功能，例如代码缩略图，[Python](https://baike.baidu.com/item/Python" \t "_blank)的插件，代码段等。还可自定义键绑定，菜单和工具栏。Sublime Text 的主要功能包括：拼写检查，书签，完整的 Python API ， Goto 功能，即时项目切换，多选择，多窗口等等。Sublime Text 是一个跨平台的编辑器，同时支持[Windows](https://baike.baidu.com/item/Windows" \t "_blank)、[Linux](https://baike.baidu.com/item/Linux" \t "_blank)、[Mac OS X](https://baike.baidu.com/item/Mac%20OS%20X" \t "_blank)等操作系统。作为主流前端开发编辑器，它不仅体积较小，运行速度快并且文本功能强大，同时支持编译功能且可在控制台看到输出，内嵌python解释器支持插件开发以达到可扩展目的和内置Package Control——可支持的大量插件可通过其进行管理。

## 2.2.1 基于chrome扩展程序React Developer Tools的调试

React Developer Tools是一款由facebook开发的有用的[Chrome 浏览器扩展](http://www.cnplugins.com/" \t "_blank)，可以通过 Chrome Web存储获取。使用 Chrome Devtools 进行调试时，可以查看应用程序的 React 组件分层结构，而不是更加神秘的浏览器 DOM 表示。添加react developer tools到chrome，是对chrome开发工具的React调试工具。React的开发工具是开源Chrome DevTools延伸反应的JavaScript库。它允许你检查React在Chrome开发者工具组件的层次结构（原名WebKit Web Inspector）。你会得到新的标签要求在你的Chrome DevTools反应。这表明你的根反应组件在页面渲染，以及他们最终渲染组件。

插件安装成功后，浏览器的右上方出现现图示的log，用户如果想要调试的话就点击log就可以了。通过选择树中的一个组件，你可以在右边的面板中检查和编辑它的当前的道具和状态。在面包屑，你可以检查所选组件，创建它的组件，创建一个组件，等等。如果您使用常规元素选项卡检查页面上的一个反应元素，然后切换到“React”选项卡，该元素将在“React树”中自动选择。

如图2.1所示显示了安装了 React Developer Tools 的 example3 组件分层结构外观。

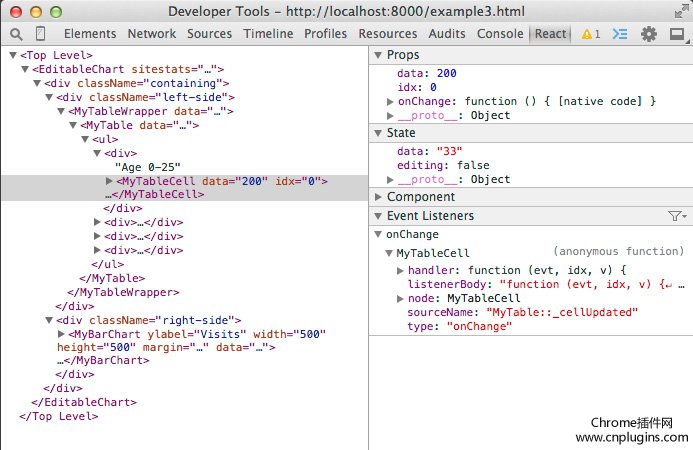


图2.1 React Developer Tools 的 example3 组件分层结构外观图

类似地，如图2.2.2，如果在一个组件的渲染阶段有一个断点，将在“React”选项卡中自动选择。这允许你在渲染树中一步一步看到一个组件如何影响另一个。通过打破错误，你可以很容易地找到哪些组件在渲染过程中抛出一个错误，和什么样的道具导致它

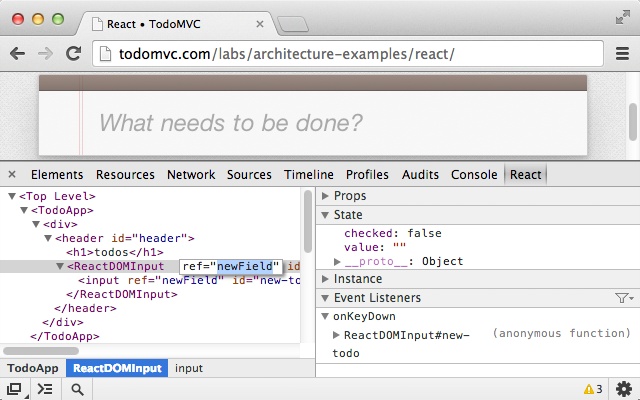


图2.2 错误断点调试

## 2.2 Node.js和npm介绍

## 2.2.1 Node.js的介绍和开发优势

Node.js是一个Javascript运行环境(runtime environment)，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装。Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。

V8引擎执行Javascript的速度非常快，性能非常好。[1]  Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用[事件驱动](https://baike.baidu.com/item/%E4%BA%8B%E4%BB%B6%E9%A9%B1%E5%8A%A8" \t "_blank)， 非阻塞[I/O](https://baike.baidu.com/item/I%2FO/84718" \t "_blank) 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。

V8引擎本身使用了一些最新的编译技术。这使得用Javascript这类[脚本语言](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80" \t "_blank)编写出来的代码运行速度获得了极大提升，又节省了开发成本。对性能的苛求是Node的一个关键因素。 Javascript是一个[事件驱动](https://baike.baidu.com/item/%E4%BA%8B%E4%BB%B6%E9%A9%B1%E5%8A%A8" \t "_blank)语言，Node利用了这个优点，编写出可扩展性高的服务器。Node采用了一个称为“事件循环(event loop）”的架构，使得编写可扩展性高的服务器变得既容易又安全。提高服务器性能的技巧有多种多样。Node选择了一种既能提高性能，又能减低开发复杂度的架构。这是一个非常重要的特性。并发编程通常很复杂且布满地雷。Node绕过了这些，但仍提供很好的性能。

Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（[寻址](https://baike.baidu.com/item/%E5%AF%BB%E5%9D%80" \t "_blank)并检索文件），硬盘准备好的时候非阻塞接口会通知Node。该模型以可扩展的方式简化了对慢资源的访问， 直观，易懂。尤其是对于熟悉[onmouseover](https://baike.baidu.com/item/onmouseover" \t "_blank)、onclick等[DOM](https://baike.baidu.com/item/DOM/50288" \t "_blank)事件的用户，更有一种似曾相识的感觉。

虽然让Javascript运行于服务器端不是Node的独特之处，但却是其一强大功能。不得不承认，浏览器环境限制了我们选择编程语言的自由。任何服务器与日益复杂的浏览器客户端应用程序间共享代码的愿望只能通过Javascript来实现。虽然还存在其他一些支持Javascript在服务器端 运行的平台，但因为上述特性，Node发展迅猛，成为事实上的平台。

### 2.2.2 npm包管理工具的使用介绍

NPM的全称是Node Package Manager，是一个NodeJS包管理和分发工具，已经成为了非官方的发布Node模块（包）的标准。

Nodejs自身提供了基本的模块，但是开发实际应用过程中仅仅依靠这些基本模块则还需要较多的工作。幸运的是，Nodejs库和框架为我们提供了帮助，让我们减少工作量。但是成百上千的库或者框架管理起来又很麻烦，有了NPM，可以很快的找到特定服务要使用的包，进行下载、安装以及管理已经安装的包。

它是世界上最大的软件注册表，每星期大约有 30 亿次的下载量，包含超过 600000 个 包（package） （即，代码模块）。来自各大洲的开源软件开发者使用 npm 互相分享和借鉴。包的结构使您能够轻松跟踪依赖项和版本。

npm 由三个独立的部分组成:网站，注册表（registry），命令行工具（CLI）。[网站](https://npmjs.com/) 是开发者查找包（package）、设置参数以及管理 npm 使用体验的主要途径。注册表 是一个巨大的数据库，保存了每个包（package）的信息。[CLI](https://docs.npmjs.com/cli/npm) 通过命令行或终端运行。开发者通过 CLI 与 npm 打交道。

## 2.3打包工具webpack介绍

现今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法

* 模块化，让我们可以把复杂的程序细化为小的文件;
* 类似于TypeScript这种在JavaScript基础上拓展的开发语言：使我们能够实现目前版本的JavaScript不能直接使用的特性，并且之后还能转换为JavaScript文件使浏览器可以识别；
* Scss，less等CSS预处理器

这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常繁琐的，这就为WebPack类的工具的出现提供了需求。

webpack 是一个现代 JavaScript 应用程序的静态模块打包器(module bundler)。当 webpack 处理应用程序时，它会递归地构建一个依赖关系图(dependency graph)，其中包含应用程序需要的每个模块，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其转换和打包为合适的格式供浏览器使用。

## 2.4代码规范及ESLint介绍使用

ESLint是一个用来识别 ECMAScript 并且按照规则给出报告的代码检测工具，使用它可以避免低级错误和统一代码的风格。ESLint被设计为完全可配置的，主要有两种方式来配置ESLint：

* 在注释中配置：使用JavaScript注释直接把配置嵌入到JS文件中。
* 配置文件：使用下面任一的文件来为全部的目录和它的子目录指定配置信息。
  + javascript：使用.eslintrc.js文件并导出一个包含配置的对象。
  + YAML：.eslintrc.yaml或者.eslintrc.yml
  + JSON：.eslintrc.json，并且此文件允许使用JS形式的注释
  + 废弃的用法：.eslintrc，此文件可以是JSON或者YAML
  + package.json：在package.json文件中创建eslintConfig属性，所有的配置包含在此属性中。

这些文件的优先级则是按照以上出现的顺序（.eslintrc.js > .eslintrc.yaml > .eslintrc.yml > .eslintrc.json > .eslintrc > package.json）。

可以被配置的信息主要分为3类：

* Environments：你的 javascript 脚步将要运行在什么环境（如：nodejs，browser，commonjs等）中。
* Globals：执行代码时脚步需要访问的额外全局变量。
* Rules：开启某些规则，也可以设置规则的等级。

## 2.5 MongoDB数据库介绍

MongoDB  是一个基于分布式文件存储的数据库。由[C++](https://baike.baidu.com/item/C%2B%2B" \t "_blank)语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。

MongoDB  是一个介于[关系数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93" \t "_blank)和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似[json](https://baike.baidu.com/item/json" \t "_blank)的[bson](https://baike.baidu.com/item/bson" \t "_blank)格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立[索引](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95" \t "_blank)。

它的特点是高性能、易部署、易使用，存储数据非常方便

主要功能特性有：

\*面向集合存储，易存储对象类型的数据。

\*模式自由。

\*支持动态[查询](https://baike.baidu.com/item/%E6%9F%A5%E8%AF%A2" \t "_blank)。

\*支持完全索引，包含内部对象。

\*支持查询。

\*支持复制和故障恢复。

\*使用高效的二进制数据存储，包括大型对象（如视频等）。

\*自动处理碎片，以支持云计算层次的扩展性。

\*支持[RUBY](https://baike.baidu.com/item/RUBY" \t "_blank)，[PYTHON](https://baike.baidu.com/item/PYTHON" \t "_blank)，[JAVA](https://baike.baidu.com/item/JAVA" \t "_blank)，[C++](https://baike.baidu.com/item/C%2B%2B" \t "_blank)，[PHP](https://baike.baidu.com/item/PHP" \t "_blank)，[C#](https://baike.baidu.com/item/C%23" \t "_blank)等多种语言。

\*文件存储格式为BSON（一种JSON的扩展）。

\*可通过[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C" \t "_blank)访问。

# 第3章系统分析与设计

## 3.1 系统需求分析

本系统是基于AI语音识别的websocket聊天室，主要的功能是通过AI语音识别聊天内容并采用websocket传输方式，进行实时数据交流，并使用机器人将语音识别的内容进行智能回复。

主要的业务是

1. 调用HTML5标准录音接口对接收到的音频进行接收处理，分离，转换，解码成可以进行语音识别的音频内容。
2. 调用语音识别接口，对转换成功的音频文件进行语音识别。
3. 采用websocket进行实时的数据传输，保证数据传输的实时性。
4. 调用机器人接口对语音识别成功的内容进行智能回复。

系统的功能需求是

1. 登录注册页面的成功实现
2. 录音功能的实现
3. 聊天室功能的实现
4. 系统UI界面友好，交互性好，易于操作

## 3.2系统概要设计

### 3.2.1 系统结构设计

系统结构图，如图3-1所示：

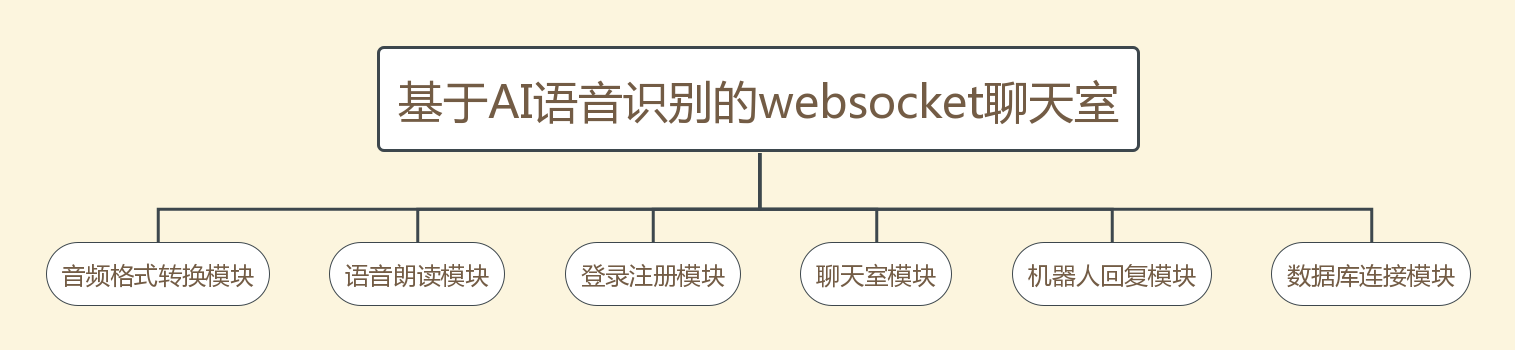


图3-1 系统结构图

系统登陆注册流程图，如图3-2：

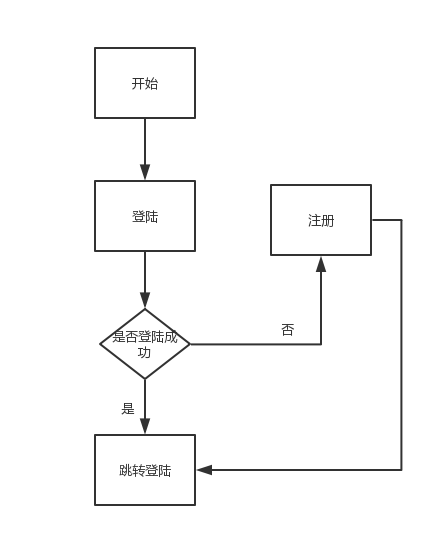


图3.2 登陆注册流程图

# 第4章 系统的具体实现

## 4.1 前端的具体实现

### 4.1.1 音频转换模块

获取HTML5音频对象：

var context = new AudioContext();

创建音频输入对象：

var audioInput = context.createMediaStreamSource(stream);

创建录音对象：

var recorder = context.createScriptProcessor(4096, 1, 1);

初始化音频转换变量：

size: 0, //录音文件长度

buffer: [], //录音缓存

inputSampleRate: context.sampleRate, //输入采样率 默认为 44.1k

inputSampleBits: 16, //输入采样数位 8, 1， 百度语音识别: 原始语音的录音格式目前只支持评测 8k/16k 采样率 16bit 位深的单声道语音

outputSampleRate: 16000, //输出采样率

oututSampleBits: 16, //输出采样数位 8, 16

对接收到的音频进行压缩合并的函数：

compress: function () { //合并压缩

//合并

var data = new Float32Array(this.size);

var offset = 0;

for (var i = 0; i < this.buffer.length; i++) {

data.set(this.buffer[i], offset);

offset += this.buffer[i].length;

}

//压缩

var compression = parseInt(this.inputSampleRate / this.outputSampleRate , 10);

var length = data.length / compression;

var result = new Float32Array(length);

var index = 0, j = 0;

while (index < length) {

result[index] = data[j];

j += compression;

index++;

}

return result;

},

对音频进行编码的函数：

encodeWAV: function () {

var sampleRate = Math.min(this.inputSampleRate, this.outputSampleRate);

var sampleBits = Math.min(this.inputSampleBits, this.oututSampleBits);

var bytes = this.compress();

var dataLength = bytes.length \* (sampleBits / 8);

var buffer = new ArrayBuffer(44 + dataLength);

var data = new DataView(buffer);

var channelCount = 1;//单声道

var offset = 0;

var writeString = function (str) {

for (var i = 0; i < str.length; i++) {

data.setUint8(offset + i, str.charCodeAt(i));

}

}

// 资源交换文件标识符

writeString('RIFF'); offset += 4;

// 下个地址开始到文件尾总字节数,即文件大小-8

data.setUint32(offset, 36 + dataLength, true); offset += 4;

// WAV文件标志

writeString('WAVE'); offset += 4;

// 波形格式标志

writeString('fmt '); offset += 4;

// 过滤字节,一般为 0x10 = 16

data.setUint32(offset, 16, true); offset += 4;

// 格式类别 (PCM形式采样数据)

data.setUint16(offset, 1, true); offset += 2;

// 通道数

data.setUint16(offset, channelCount, true); offset += 2;

// 采样率,每秒样本数,表示每个通道的播放速度

data.setUint32(offset, sampleRate, true); offset += 4;

// 波形数据传输率 (每秒平均字节数) 单声道×每秒数据位数×每样本数据位/8

data.setUint32(offset, channelCount \* sampleRate \* (sampleBits / 8), true); offset += 4;

// 快数据调整数 采样一次占用字节数 单声道×每样本的数据位数/8

data.setUint16(offset, channelCount \* (sampleBits / 8), true); offset += 2;

// 每样本数据位数

data.setUint16(offset, sampleBits, true); offset += 2;

// 数据标识符

writeString('data'); offset += 4;

// 采样数据总数,即数据总大小-44

data.setUint32(offset, dataLength, true); offset += 4;

// 写入采样数据

if (sampleBits === 8) {

for (let i = 0; i < bytes.length; i++ , offset++) {

var s = Math.max(-1, Math.min(1, bytes[i]));

var val = s < 0 ? s \* 0x8000 : s \* 0x7FFF;

val = parseInt(255 / (65535 / (val + 32768)), 10);

data.setInt8(offset, val, true);

}

} else {

for (let i = 0; i < bytes.length; i++ , offset += 2) {

let s = Math.max(-1, Math.min(1, bytes[i]));

data.setInt16(offset, s < 0 ? s \* 0x8000 : s \* 0x7FFF, true);

}

}

return new Blob([data], { type: 'audio/wav' });

}

开始录音，并调用HTML5音频输入函数，并连接recorder，进行音频录入的开始：

this.start = function () {

audioInput.connect(recorder);

recorder.connect(context.destination);

}

停止录音：

this.stop = function () {

recorder.disconnect();

}

获取音频文件：

this.getBlob = function () {

this.stop();

return audioData.encodeWAV();

}

回放：

this.play = function (audio) {

audio.src = window.URL.createObjectURL(this.getBlob());

}

音频采集：

recorder.onaudioprocess = function (e) {

audioData.input(e.inputBuffer.getChannelData(0));

}

进行HTML5音频采集初始化操作：

H5Recorder.init = function (callback) {

navigator.mediaDevices.getUserMedia({ audio: true })

.then(function (stream) {

var rec = new H5Recorder(stream);

callback(rec);

})

.catch(function (err) {

console.error(err)

})

}

绑定到当前window的全局变量，利于react中进行组件初始化调用：

window.H5Recorder = H5Recorder;

### 4.1.2 语音识别模块

在react组件中设置语言识别状态：

setState (status) {

if( status === 'recording') {

this.recording = true

this.recognizing = false

} else if( status === 'recognizing') {

this.recognizing = true

} else {

this.recording = false

this.recognizing = false

}

}

发送语言到后台服务器进行语音识别：

emitMes() {

var \_this = this;

if (\_this.recognizing) {

return

}

if (! \_this.recording) {

window.H5Recorder.init(function (rec) {

\_this.recorder = rec

\_this.recorder.start()

})

\_this.setState('recording')

} else {

let buffer = \_this.recorder.getBlob()

this.socket.emit('chat message', buffer)

\_this.setState('recognizing')

}

### 4.1.3 语音朗读模块

### HTML5语音Web Speech API: HTML5中和Web Speech相关的API实际上有两类，一类是“语音识别(Speech Recognition)”，另外一个就是“语音合成(Speech Synthesis)”，这两个名词听上去很高大上，实际上指的分别是“语音转文字”，和“文字变语音”。

“语音识别”和“语音合成”看上去像是正反两方面，应该带有镜面气质，实际上，至少从兼容性来看，两者并无法直接对等。

“语音识别(Speech Recognition)”目前的就Chrome浏览器和死忠小弟Opera浏览器默认支持，并且需要webkit私有前缀，如图4-1：（本文不调用此Web Speech API，语音识别有另外的API可寻。）

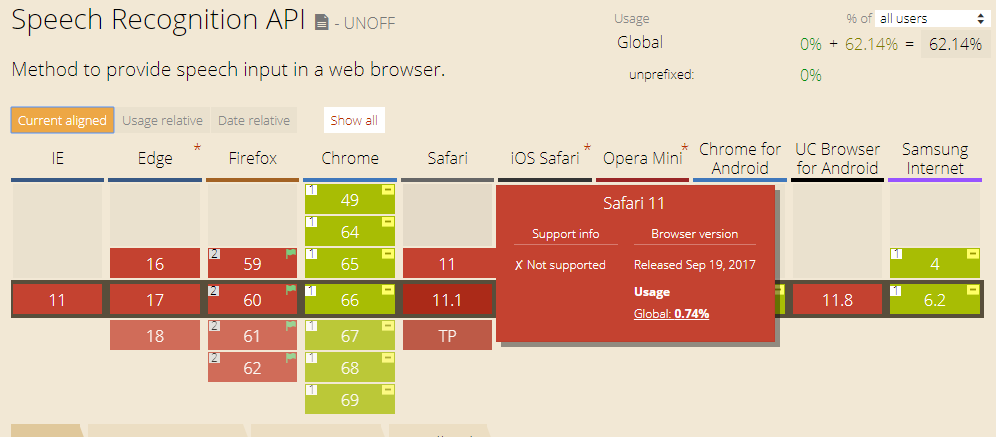


图4-1 Speech Recognition兼容性

“语音合成(Speech Synthesis)”的兼容性要好上太多了，如图4-2：

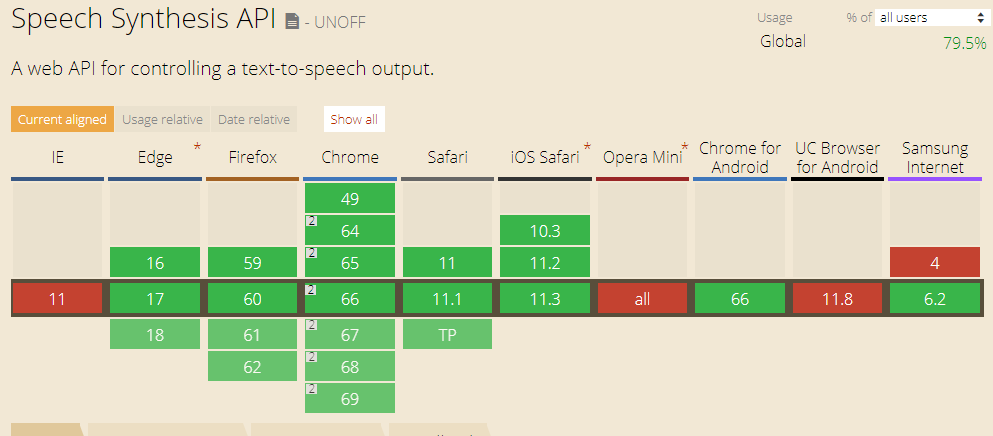


图4-2 Speech Synthesis兼容性

语音朗读模块主要调用Web Speech API的语音合成API，进行语音的朗读：

function synthVoice(text) {

// 声明speechSynthesis对象

const synth = window. speechSynthesis

// 新建语音合成实例

const utterance = new SpeechSynthesisUtterance()

// 设置朗读语音为中文

utterance.lang = 'zh';

utterance.text = text

synth.speak(utterance)

}

并且语音合成需要系统内置TTS模块，如果当前操作系统为精简版操作系统或者系统未安装TTS模块，将导致语音合成失败。

### 4.1.4 登录注册模块

登录注册页面背景采用css3技术绘制动画：

@keyframes change-bac {

50% {

background-image: url(../imgs/login1.jpg) ;

background-size:100% 100%;

background-attachment: fixed;

background-position: center center;

background-repeat: no-repeat;

}

100% {

background-image: url(../imgs/login2.jpg) ;

background-size:100% 100%;

background-attachment: fixed;

background-position: center center;

background-repeat: no-repeat;

}

}

登录页面和注册页面通用一个模块：

class Login extends Component {

render() {

return (

<div className="background">

<div className="login">

<LoginForm history = { this.props.history } />

</div>

</div>

);

}

}

export default Login;

class Login extends Component {

render() {

return (

<div className="background">

<div className="register">

<RegisterForm history = { this.props.history } />

</div>

</div>

);

}

}

export default Login;

登录注册两个页面使用ant.design框架的form表单

采用fetch模块获取数据，并向后台服务器发送数据获取相应：

fetch('http://localhost:8000/login', {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify(values)}).then((res) => {

return res.json();

}).then((res) => {

// 返回响应成功 注册成功

if (res.success) {

console.log(res)

openNotification(res.info + '将自动跳转');

setTimeout(() => \_this.props.history.push('/talking'), 2000);

}else {

openNotification(res.info);

setTimeout(() => \_this.props.history.push('/register'), 2000);

}

});

### 4.1.5 聊天室模块

聊天室模块使用react + ant.design设计，前端在模块初始化时向后台获取用户的基本信息，包括聊天室的聊天人数，聊天的历史记录，聊天的搜索内容等等，父组件和子组件之间的通信采用react的props传输，内部状态通过this.state保存：

constructor(props) {

super(props);

this.state = {

talker: ''

}

}

changeTalker(talker) {

console.log(talker);

this.setState({talker: talker})

// this.talker = talker;

}

## 4.2 后端express + websocket设计

### 4.1.1 websocket设计模块

websocket模块采用koa-socket进行开发，它是搭载在koa框架上的一款socket模块，使用koa-socket能实现websocket的全部功能。

开始先在koa上装载app:

io.attach(app)

然后开始监听：

io.on('connection', (socket) => {

console.log('a user connected')

})

再对事件进行监听，以便能及时的收到用户的消息：

io.on('chat message', (socket) => {})

### 4.1.2 语音识别接口模块

语音识别使用baidu-ai的SDK，通过调用baidu-ai的接口，可以很方便的进行语音的智能识别并返回可处理的错误信息。

首先使用之前需要在baidu-ai注册当前的app\_id，通过app\_id可以拿到当前app的api\_key和secret\_key，然后在连接语音识别时注册当前app的客户端：

const client = new AipSpeech(APP\_ID, API\_KEY, SECRET\_KEY)

然后上传到baidu-ai进行语音识别：

client.recognize(voiceBuffer, 'wav', 16000, { lan: 'zh' })

.then(function (res) {

console.log('This is what you say')

console.log(res)

if (res.err\_no === 0) {

let text = res.result[0]

SendChat(text)

.then((res) => {

console.log('This is return message')

console.log(res.text)

console.log(res)

io.broadcast('bot reply', { say: text, replay: res.text })

})

.catch((err) => {

console.error(err)

})

### 4.1.3 机器人回复模块

机器人回复是语音识别聊天室的另一个核心功能，主要调用图灵机器人接口，图灵机器人接口是通过自己在图灵机器实验室自行管理的一个项目实例，语音识别的内容可以通过系统自动回复或者选择自己进行模式训练。

使用机器人接口需要在图灵机器人网站管理自己的项目实例，注册项目所需的一些了方法，key，info，和userid。

首先定义图灵机器人配置文件：

const request = require('request-promise')

function SendChat (text) {

const textTpl = {

'key': '3cadbeb78b354734afc88bd0a13daa26',

'info': text,

'userid': '205216'

}

let options = {

method: 'POST',

uri: 'http://www.tuling123.com/openapi/api',

body: textTpl,

json: true

}

return request(options)

}

module.exports = SendChat

然后在导出的SendChat的方法中使用接口获取机器人实例结果：

SendChat(text)

.then((res) => {

console.log(res.text)

console.log(res)

})

### 4.1.4 登录注册模块

登录注册模块主要定义在express中的route文件夹中，然后在导出到express服务器中进行路由注册。

登录注册后台服务器主要进行用户的注册和登录使用，其中对用户名密码等关键性信息进行加密处理，加密处理主要调用npm的crypto模块进行密码加密，保证用户信息安全。

加密模式采用md5加密：

password : md5.update(req.body.password).digest('base64')；

登录注册使用的方法主要包括：

// 查询用户方法

async function findUser (db, collectionName,query) {

const collection = db.collection(collectionName);

const result = await collection.find(query).count()

return result;

}

// 插入user方法

async function insertUserInfo (db, collectionName, doc) {

const collection = db.collection(collectionName);

const { result } = await collection.insertOne(doc);

return result;

}

### 4.1.5 聊天室模块

聊天室模块是在用户登录注册成功后进行跳转到的页面时需要使用的后台模块，在用户进行聊天室模块后，会自动向后台服务器请求用户信息相关的信息，包括聊天人的消息内容，机器人回复的消息内容的等等。

### 4.1.6 数据库连接模块

数据库主要使用NO-SQL式的MongoDB数据库，在express服务框架中连接数据库时主要使用mongodb数据库模块。

首先设置数据库配置：

const url = 'mongodb://localhost:27017';

const dbName = 'talking-room';

const collectionName = 'users'；

初始化数据库：

const MongoClinet = require('mongodb').MongoClient;

进行数据库连接:

MongoClinet.connect(url, async function (err, client) {

console.log('connect db successfully');

const db = await client.db(dbName);

}

关闭数据库操作：

client.close();

console.log('关闭数据库') ;

# 

# 

# 第5章 系统测试

系统测试，英文是System Testing。是将已经确认的软件、计算机硬件、外设、网络等其他元素结合在一起，进行信息系统的各种组装测试和确认测试，系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。系统测试发现问题之后要经过调试找出错误原因和位置，然后进行改正。是基于系统整体需求说明书的黑盒类测试，应覆盖系统所有联合的部件。对象不仅仅包括需测试的软件，还要包含软件所依赖的硬件、外设甚至包括某些数据、某些支持软件及其接口等

## 5.1**软件测试的目的**

软件测试是程序的一种执行过程，目的是尽可能发现并改正被测试软件中的错误，提高软件的可靠性。它是软件生命周期中一项非常重要且非常复杂的工作，对软件可靠性保证具有极其重要的意义。在目前形式化方法和程序正确性证明技术还无望成为实用性方法的情况下，软件测试在将来相当一段时间内仍然是软件可靠性保证的有效方法。软件工程的总目标是充分利用有限的人力和物力资源，高效率、高质量地完成软件开发项目。不足的测试势必使软件带着一些未揭露的隐藏错误投入运行，这将意味着更大的危险让用户承担。过度测试则会浪费许多宝贵的资源。到测试后期，即使找到了错误，然而付出了过高的代价。E.W.Dijkstra的一句名言说明了这一道理：“程序测试只能表明错误的存在，而不能表明错误不存在。”可见，测试是为了使软件中蕴涵的缺陷低于某一特定值，使产出、投入比达到最大。

## 5.2**软件测试的方法**

1．可用性测试，英文是Practical Usability Testing。可用性测试是对“用户友好性”的测试。显然这是主观的，且将取决于目标最终用户或客户。用户面谈、调查、用户对话的录象和其他一些技术都可使用。程序员和测试员通常都不宜作可用性测试员。

2．压力测试，英文是Stress Testing。和负载测试差不多。压力测试是一种基本的质量保证行为，它是每个重要软件测试工作的一部分。压力测试的基本思路很简单：不是在常规条件下运行手动或自动测试，而是在计算机数量较少或系统资源匮乏的条件下运行测试。通常要进行压力测试的资源包括内部内存、CPU 可用性、磁盘空间和网络带宽等。一般用并发来做压力测试。

3．负载测试，英文是Load testing。负载测试是测试一个应用在重负荷下的表现。例如测试一个 Web 站点在大量的负荷下，何时系统的响应会退化或失败，以发现设计上的错误或验证系统的负载能力。在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。

负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征，例如，响应时间、事务处理速率和其他与时间相关的方面。

4．健全测试，英文是Sanity testing。健全测试是指一个初始化的测试工作，以决定一个新的软件版本测试是否足以执行下一步大的测试能力。例如，如果一个新版软件每5分钟与系统冲突，使系统陷于泥潭，说明该软件不够“健全”，不具备进一步测试的条件。

## 5.3测试方案

1．单元测试方案，在src中的test文件夹中新建前端及后端测试文件：

a) 新建测试模块，进行单元测试，主要测试模块的连通性和完整性

b) 测试算法的准确性，采用多种数据测试的方式，尽量覆盖多的测试方案

2. 白盒测试，主要测试系统相关逻辑：

a) 测试登录注册逻辑判断及其跳转问题

b) 测试数据库写入数据逻辑

3. 黑盒测试，测试系统相关功能的具体实现情况：

1. 测试登录注册功能是否完整实现
2. 测试语音识别功能的正确性
3. 测试机器人聊天功能实现
4. 测试语音朗读功能的实现

## 5.4测试用例

主要列出登录注册模块和聊天室模块的功能测试点 （单元测试主要运行在srcc/test文件中）见表5-1：

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 测试用例描述 | 操作过程及数据 | 预期结果 |
| 登录注册 | 输入注册数据 | 输入需要注册的用户名密码 | 正常注册 |
| 输入已注册过的用户名 | 输入已注册过的用户名密码 | 提示已经注册过用户名 |
| 输入登录数据 | 输入错误的登录用户名密码 | 登录错误 |
| 输入未注册的登录数据 | 输入暂未注册的用户名密码 | 提示未注册并跳转注册 |
| 聊天室模块 | 输入声音较低的音频 | 麦克风录音时声音较低 | 返回错误，音频质量太差 |
| 输入声音较高的音频 | 麦克风录音时声音较高 | 返回错误，音频质量太差 |
| 输入正常的音频 | 麦克风录音时正常录音 | 音频质量良好，语音识别正确 |

## 

## 5.5 测试结论

从单元测试和黑盒测试结果中，发现系统能按照预期效果运行成功，语音识别和语音朗读，机器人智能回复等结果对系统的支持良好

# 总结

基于AI语音识别的websocket聊天室从最初的探寻HTML5接口开始，到一步一步摸索浏览器接口，慢慢琢磨语音识别SDK开始，到完整的做出websocket聊天室，总共用了四个多月时间，前期HTML5语音朗读的接口不通，TTS引擎的中断给系统的开发增加了很多难度，中期语音识别接口和机器人回复模块的不清晰让我有时候感到自己系统选择的正确性。索性最后还是一点一点的坚持下来了，仔细的上各个网站找接口，去技术论坛进行问答等等。最终还是实现了整个系统。

基于AI语音识别的websocket聊天室核心模块为语音识别功能，机器人回复功能和聊天室功能，采用websocket技术，响应及时，数据交互清晰。前端的实现采用社区流行的react + ant.design框架，结构清楚明了，交互性可扩展性良好。

# 

# 参考文献

1. 胡崧;HTML 从入门到精通 [M].北京:中国青年出版社, 2007
2. 刘智勇; JavaScript 开发技术大全 [M].北京:清华 大学出版社, 2009.
3. 吴晓东;王鹏;;[Html5的通信机制及效率的研究](http://www.cnki.com.cn/Article/CJFDTOTAL-CGJM201104045.htm" \t "_blank)[J];长春理工大学学报(自然科学版);2011年04期
4. 吴晟;苏庆堂;罗斌;赵莉楠;蔡灿民;;[基于Socket和多线程技术的并发服务器的研究](http://www.cnki.com.cn/Article/CJFDTOTAL-KMLG200604009.htm" \t "_blank)[J];昆明理工大学学报(理工版);2006年04期
5. 李鑫;[Web服务器性能优化研究与实现](http://cdmd.cnki.com.cn/Article/CDMD-10217-1012266098.htm" \t "_blank)[D];哈尔滨工程大学;2011年
6. 薛真真;[基于服务器推送和事件流处理技术的实时Web系统研究](http://cdmd.cnki.com.cn/Article/CDMD-10335-2008066040.htm" \t "_blank)[D];浙江大学;2008年
7. 周巍;[基于请求策略和状态切换的服务器推送技术研究](http://cdmd.cnki.com.cn/Article/CDMD-10511-2009159578.htm" \t "_blank)[D];华中师范大学;2009年
8. 邱涌;;[基于局域网的数据推送](http://www.cnki.com.cn/Article/CJFDTOTAL-GZDN201101030.htm" \t "_blank)[J];电脑与电信;2011年01期
9. 罗大晖;陈娟;;[基于HTML5的Web离线应用研究与实现](http://www.cnki.com.cn/Article/CJFDTOTAL-JYRJ201212076.htm" \t "_blank)[J];计算机应用与软件;2012年12期
10. 郑强;徐国胜;;[Websocket在服务器推送中的研究](http://cpfd.cnki.com.cn/Article/CPFDTOTAL-TTTT201208001079.htm" \t "_blank)[A];第九届中国通信学会学术年会论文集[C];2012年
11. Li Deng, Dong Yu, “Deep Learning forSignal and Information Processing,” Microsoft Research, 2013.
12. RamanArora and Karen Livescu, “Multi-view learning with supervision for transformedbottleneck features,” in Proc. ICASSP, 2014.
13. Yuan Liu ;Tianfan Fu ;Yuchen Fan;Yanmin Qian ;Kai Yu, Speaker verification with deep features[J], 2014.
14. Improving HMM/DNN in ASR ofunder-resourced languages using probabilistic sampling 2015.
15. ZHANG Qingqing,LIU Yong,WANGZhichao,PAN Jielin,YAN Yonghong, The Application of Convolutional NeuralNetwork in Speech Recognition[J],2014.

# 

# 致谢

光阴似箭，一转眼已是快离开的日子。在校的这几年时间里很感谢学校老师们对我的淳淳教诲，是学校老师们教会了我们勤奋学习，诚实做人，踏实做事，以宽容之心面对生活。指引着我们沿着正确方向前进。在点滴汇聚中使我逐渐形成正确、成熟的人生观、价值观。

值此论文顺利完成之际，特别要感谢我的指导老师宋琦老师，宋琦老师给予我很大的帮助。宋琦老师治学严谨，学识渊博，为我营造了一种良好的精神氛围。在论文的完成过程中，从选题到开题报告，从写作提纲到一遍又一遍地指出每稿中的具体问题，严格把关，循循善诱。同时感谢宋琦老师在我学习期间给予的帮助。

深深的感谢呵护我成长的父母。每当我遇到困难的时候，父母总是第一个给我鼓励的人。回顾20多年来走过的路，每一个脚印都浸满着他们无私的关爱和谆谆教诲，11年的在外求学之路，寄托着父母对我的殷切期望。他们在精神上和物质上的无私支持，坚定了我追求人生理想的信念。父母的爱是天下最无私的最宽厚的爱。大恩无以言报，惟有以永无止境的奋斗，期待将来辉煌的事业让父母为之骄傲。我亦相信自己能达到目标。

另外感谢答辩组的老师，谢谢老师们对我的耐心指导和细心指正。

最后，再次向所有关心我的亲人、师长和朋友们表示深深的谢意。在即将离校之际，祝愿所有老师和同学们健康快乐。