School of Data and Computer Science, Sun Yat-sen University

AIDO 项目设计报告

陈鸿峥 刘学海 刘佳荣

2020 年 1 月 2 日

1 概览

本项目的所有源代码公开于 https://github.com/reacherhai/AIDO

1.1 项目要求

本项目需要我们设计一个数据库应用，采用客户端/服务器的结构，具体要求如下：

- 有完整的前后端架构（前端界面 + 后台数据库）
- 内容有实际意义且完整
- 支持对数据的增删改查操作

1.2 设计动机

我们小组做这个项目最初的动机来源于前半学期实验中做的 todolist。todolist 的作用类似于实际生活中的备忘录或是手账，用户可以在 todolist 中记录自己未来要做的事情，并且可以修改记录，或者是对记录进行标记（比如打上已经完成的标记）。通过使用这类工具，用户可以有效地管理未来的事务。

我们认为，现在的生活中，每个人都面临着繁琐的日常事务。就以学生来说，各门课都会布置作业，作业都有各自的 DDL，如果没有进行有效的管理，会导致规划时间不合理或是忘记作业等问题。所以我们认为管理个人未来事务是一种需求，我们可以针对这种需求去做一个项目。

虽然 todolist 可以达成管理个人事务这一目的，但我们认为仅仅如此是不够的，todolist 存在一些问题。todolist 中每一项记录的添加、删除、修改都要通过打字或多次操作实现，这在一定程度上增加了使用的难度，从而降低用户使用 todolist 的积极性，而这种类似于备忘录的应用如果更少地被使用，它的作用是要打折扣的。同时，todolist 的功能单一，只有备忘录这一功能，用户使用的积极性也比较低，这会带来与前一点相同的影响。

所以为了解决上述的两个问题，我们小组旨在设计一个类似于微软小冰或是小黄鸡的，能协助管理个人事务的智能机器人 AIDO。

1.3 技术栈

开发语言 JavaScript / HTML / CSS / Python / SQLite
浏览器环境 Chrome / Firefox / Safari
第三方库 jQuery / Bootstrap

1.4 项目分工

整个项目主要分为两个部分，前期准备规划以及代码的实现与测试。

前期规划阶段没有十分明显的分工，主要是三人针对 todolist 现有的问题进行分析，并进行调研，调查市场上类似产品，分析自己做的东西是否有意义。之后，进行规划，明确需要实现的需求，包括智能机器人以及事务管理两部分。再然后，进行工作的分配，并进行准备。

代码的实现与测试阶段的分工如下。

陈鸿峥：

- 前端页面的搭建与链接
- 页面的设计与美化

刘学海：

- 后端接口的设计与实现
- 设计数据库

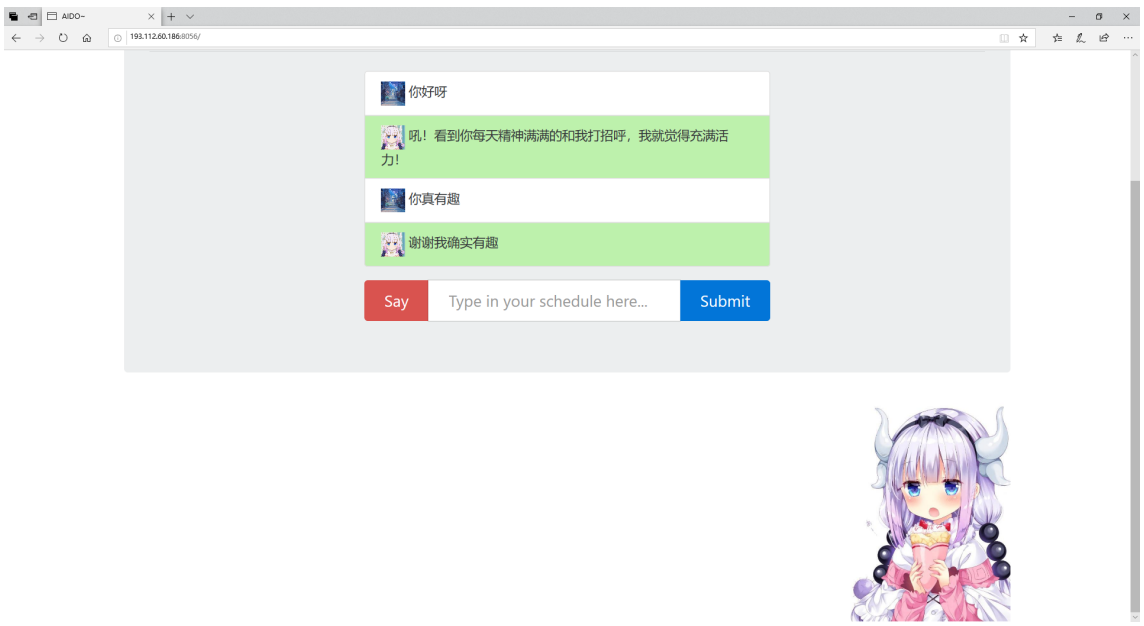
刘佳荣：

- 明确需求与实现，编写文档报告
- 对应用进行运行测试，反馈前后端

2 项目特点

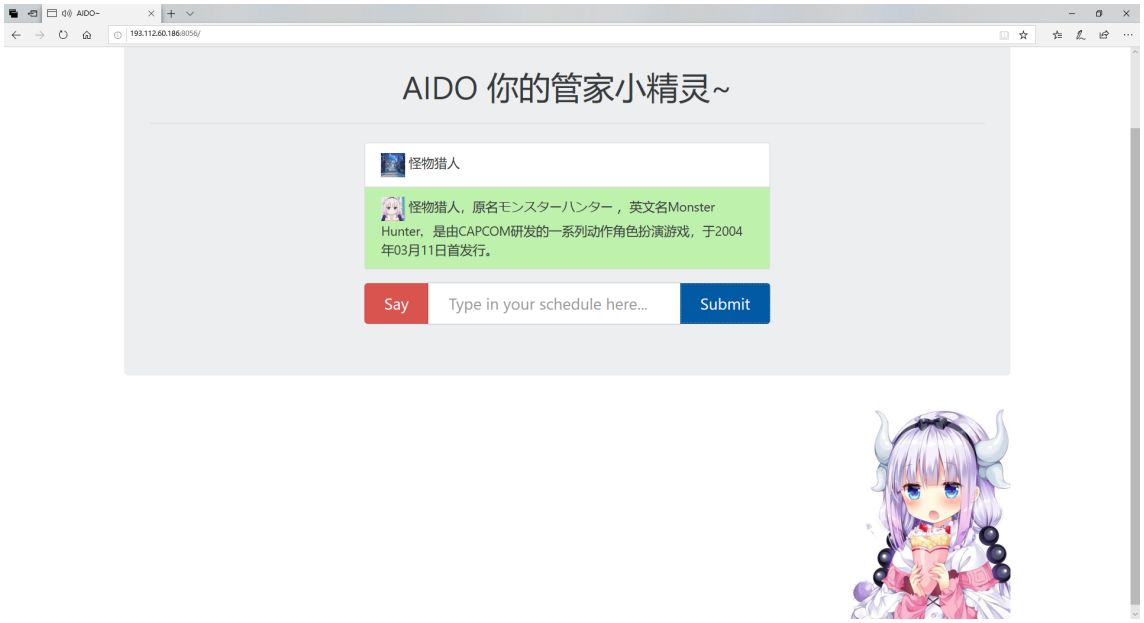
AIDO 的主要设计目的是打造一个帮助管理日常事务的智能管家，让用户能便捷、有效地管理日常事务，并提供友好的交互界面。其主要特征有：

- **聊天交互功能**：AIDO 模仿了现有的小黄鸡等应用，实现了用户与机器人之前的交互功能。



我们的机器人支持语言输入，按住 say 按钮就可以进行录音。录音文件就会保存到本地。同时，机器人也会根据语音进行相应的回答。用户可以通过与智能机器人交流，达到近似于与真人进行交流的体验。于此同时，下面的一系列功能都支持语音输入。

- **日常查询功能**：AIDO 可以根据用户输入的信息，解释相关概念。



用户可以输入想要了解的人物/事务，机器人就会为用户提供对应概念的解释，此处机器人提供的解释主要来源于百度百科。

- **添加任务功能**：AIDO 可以实现帮用户添加任务事项的功能。

用户输入日期（**** 年 * 月 * 日）+ 任务，机器人将帮用户录入 todolist，并自动计算距离 deadline 的时间。此外，支持给任务添加优先级。在任务中添加“优先级”+ 数字，奇迹人将自动检测任务的优先级并将高优先级的任务排在前面。

- **删除任务功能**：AIDO 可以实现帮助用户删除任务事项的功能。

通过输入“删除”＋任务对应的字母/数字，机器人就可以帮助用户删除对应的任务。

- **查看待办事项功能**：AIDO 允许用户查询当前剩余的待办事项。

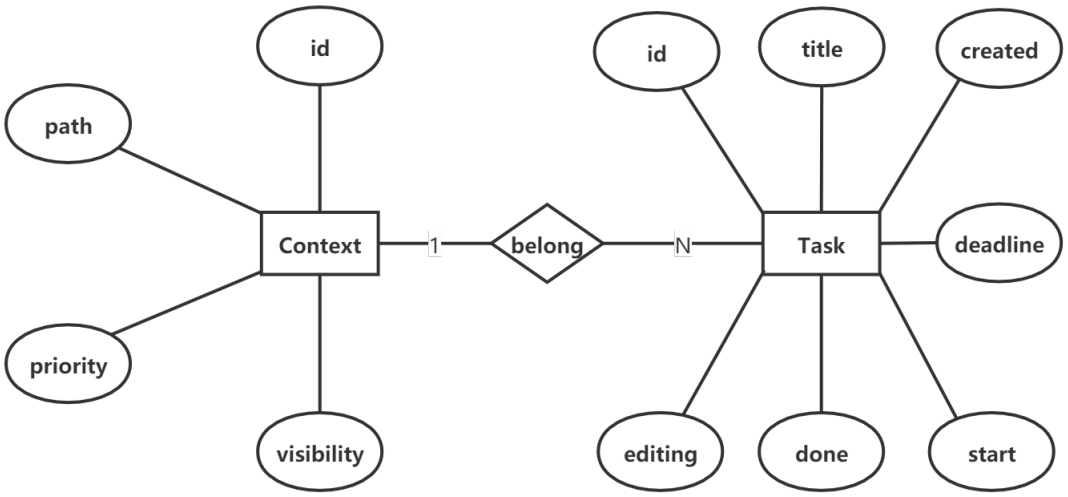
通过输入“todo”，用户可以查看现有的所有待办事项的详细信息。

我们的项目主要包含了以上功能，可以说，AIDO 包含了 todolist 的核心功能，但同时加入了智能机器人这一辅助工具，使个人事务的管理可以统一使用语音输入这一手段进行管理，增加了效率和方便性，又使得应用场景更丰富，机器人可以协助解决除事务管理外的其它问题。同时，AIDO 也提供了更为美观、友好的交互界面，给用户带来更好的体验。

3 ER 图与关系模式设计

3.1 ER 图

本项目设计的 ER 图如下所示：



3.2 关系模式与建表语句

根据以上 ER 图，我们可以建立以下关系模式：

- **Task(id, title, created, deadline, start, priority, done, context, content, editing)**

```
1  CREATE TABLE `Task` (
2          `id`      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
3          `title` TEXT NOT NULL,
4          `created`         TEXT NOT NULL DEFAULT (datetime('now')),
5          `deadline`        TEXT,
6          `start` TEXT NOT NULL DEFAULT (datetime('now')),
7          `priority`        INTEGER NOT NULL DEFAULT 1,
8          `done`    TEXT,
9          `context`         INTEGER NOT NULL REFERENCES Context(id) ON
             DELETE CASCADE
10 );
11 CREATE INDEX `DoneIndex` ON `Task` (`done`);
12 CREATE INDEX `DateCreatedIndex` ON `Task` (`created`);
13 ALTER TABLE Task ADD COLUMN `content` TEXT;
14 ALTER TABLE Task ADD COLUMN `editing` INTEGER NOT NULL DEFAULT 0;
```

- **Context(id, path, priority, visibility)**

```
CREATE TABLE `Context` (
        `id`    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
        `path`  TEXT NOT NULL UNIQUE,
        `priority`      INTEGER NOT NULL DEFAULT 1,
        `visibility`    TEXT NOT NULL DEFAULT 'normal'
);
INSERT INTO `Context` (path) VALUES ('');
CREATE INDEX `PathIndex` ON `Context` (`path` ASC);
```

4 设计细节

本节主要介绍各个部分的设计细节，包括其中涉及的代码细节与 SQL 语句。

4.1 todolist

todolist 的实现主要包含了四个模块，添加任务，删除任务，修改任务以及查询任务。

4.1.1 添加任务

todolist 中有任务的添加，以及环境 (context）的添加。

- 任务添加

```
INSERT INTO Task (title, content, context {})
VALUES (?, ?, ? {})
```

- 环境添加

```
INSERT INTO Context (path)
VALUES (?)
```

- 将一个路径中的任务复制到另一路径

```
UPDATE Task
SET context = ?
WHERE context = (
        SELECT id FROM Context
        WHERE path = ?
)
```

4.1.2 删除任务

todolist 中有对数据的删除操作，其中包括了删除任务，删除环境，以及删除满足某些要求的任务。

- 删除特定 id 的任务

```
1 DELETE FROM Task
2 WHERE id = ?
```

- 删除特定路径的环境

```
1 DELETE FROM Context
2 WHERE path LIKE ?
```

- 删除某时刻前产生，并且已完成的任务

```
1 DELETE FROM Task
2 WHERE done IS NOT NULL
3 AND created < ?
```

4.1.3 修改任务

todolist 中有对任务的修改操作，其中主要是修改任务内容，或是更改任务的标记内容（完成标记以及编辑标记）。

- 修改特定 id 的任务

```
1 UPDATE Task SET {}
2 WHERE id = ?
```

- 标注特定 id 的任务状态为完成

```
1 UPDATE Task SET done = datetime('now')
2 WHERE id = ?
3 AND done IS NULL
```

- 标记特定未处于编辑的任务为编辑状态

```
1 UPDATE Task
2 SET editing = 1
3 WHERE id = ?
4   AND editing = 0
```

- 释放某任务的编辑状态

```
1 UPDATE Task
2 SET editing = 0
3 WHERE id = ?
```

4.1.4 查询任务

todolist 中有如下的各种查询操作。

- 查询特定 id 的任务

```
1 SELECT t.*, c.path as ctx_path
2 FROM Task t JOIN Context c
3 ON t.context = c.id
4 WHERE t.id = ?
```

- 根据路径查询环境 id

```
1 SELECT id FROM Context
2 WHERE path = ?
```

- 查询特定路径的环境是否存在

```
1 SELECT 1 FROM Context
2 WHERE path = ?
```

- 查询特定路径环境包含的任务数和子环境数

```
1  SELECT COUNT(t.id)
2  FROM Task t
3  JOIN Context c
4  ON t.context = c.id
5  WHERE t.done IS NULL
6  AND c.path = ?
7  UNION ALL
8  SELECT COUNT(id)
9  FROM Context
10 WHERE path LIKE ?
```

- 查询特定路径下的所有任务（这些任务直接在该目录下，不在子目录）：

```
SELECT t.*, c.path as ctx_path
FROM Task t JOIN Context c
ON t.context = c.id
WHERE c.path = ?
  AND t.done IS NULL
  AND (c.path = ? OR c.visibility = 'normal')
  AND (datetime('now')) >= datetime(t.start)
ORDER BY
   priority DESC,
  COALESCE(
       julianday(deadline),
       julianday('9999-12-31 23:59:59')
    ) - julianday('now') ASC,
   created ASC
```

- 查询最大的任务 id

```
SELECT MAX(id)
FROM Task
```

- 查询特定路径下特定标题的任务

```
SELECT t.*, c.path as ctx_path
FROM Task t JOIN Context c
ON t.context = c.id
WHERE t.title LIKE ?
  AND c.path LIKE ?
```

4.2 对话机器人

4.3 交互界面

5 技术难点

5.1 后端设计

5.2 前端设计

6 结语

本项目至此便已圆满结束，在这个项目中确实学习到了很多在课堂上学习不到的知识，也提升了项目开发能力。我们有几个简单的感想：

- 实际的项目开发比原想的要复杂的多。对比以往有明确要求的各种大作业，这次的大作业只是给了基本的要求，要做什么项目、要怎么实现、要达到什么程度等都得自行决定。因此，这次的作业，前期的准备规划阶段对比以往的作业也尤为重要，小组内需要商讨项目内容，需要调查市场竞品，还得考虑到实际的开发时间等，所有这些都商量好之后才能进入之后的开发阶段，进行代码的实现与测试。

- 代码的复杂使实现的过程也存在困难。这次的作业要求我们实现包含前端界面和后端服务器的一个完整程序。后端包含的两个模块虽然都有比较成熟的模板，但是实际开发过程中，配置代码运行的环境，以及两部分代码的衔接，实现原有代码没有的功能等，耗费了队友的大量精力。前端界面的开发虽然在前半学期的 todolist 中有练习过，但在实际的开发中这并不足够，所以也是一边学习一边实现。同时，在接口的对接时，两位队友也进行了大量的沟通。这也让我们反省我们的开发流程是否还有优化的空间，可能在前期进行规划的时候需要对各种模块端口进行更为明确的规定。

总之，我感谢我两位队友与我的组队！面对工作量如此巨大的 Project，队友的努力，让这个项目有坚持下去并圆满完成的可能！

以及，十分感谢助教检查我们的项目，提出宝贵的建议，并不厌其烦地读完这篇冗长的报告。