KENNEDY AGUSI
FALL 2017

INDEPENDENT PROJECT

PROJECT TITLE: Meliora Scheduler

GOAL: The main goal of this project is to automate the process by which students schedule appointments with their professors and advisers.

SIGNIFICANCE OF THIS PROJECT: Over the years the students have been using either a signup sheet to schedule appointments with their professors, but with this app students can now schedule appointments using their mobile phones.

PROJECT DESCRIPTION

This project has three main components;

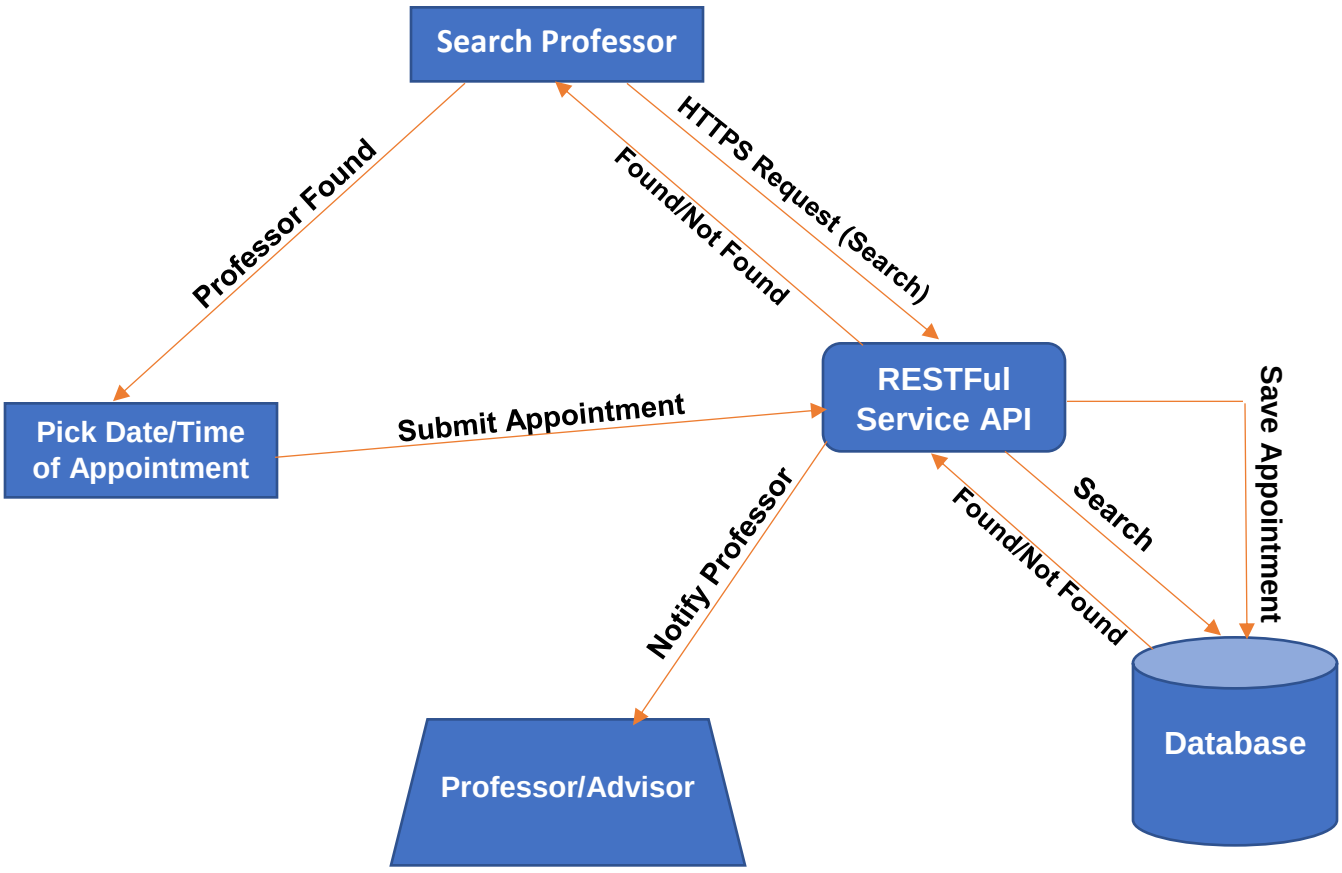
- ❖ Mobile App
- ❖ Web App
- ❖ RESTFul Service API

MOBILE APP:

The Mobile app version of Meliora Scheduler is for students use only. It has the following components**;**

- ❖ **Book Appointment Fragment:** This is used to schedule an appointment with either a professor or advisor. In this fragment, a student will first search the professor/adviser he/she wishes to book an appointment with. The search query is sent to RESTFul Service API which verifies whether the specified professor/advisor exists. If found, the API return the professor's/advisor's available schedule. A fragment will be displayed for the student to select available date/time of appointment. Once a student submit/confirms an appointment, the API processes the information and then sends a notification to the professor notifying him/her of the upcoming appointment.

Book Appointment Flow Chart



- **Appointments Fragment:** This fragment displays all currently booked appointment by a student. It also has an option for student to cancel an appointment. Once an appointment is cancelled; a notification is sent to the professor/advisor.

- **Appointment History Fragment:** This fragment displays appointment history of a student including cancelled and non-cancelled appointments.

- **Create Account Fragment:** This fragment is used to sign up for access to use Meliora Scheduler. Though this could be replaced with school's login access.

- **Login Fragment:** This fragment is used to login to Meliora Scheduler. This could also be replaced by school's login access.

- **Notification Service:** In this project, I used Google's firebase notification service. I used a free version for now but during actual deployment to app store, a paid version will be required.

WEB APP:

The web app version of Meliora Scheduler is for professors/advisors use only. It has the following components.

- **Sign up page:** This page is used to register a professor/advisor in other to grant access to Meliora Scheduler.

- **Login page:** This is used to login to Meliora Scheduler

- **Account page:** This page has three main components;

    - **Profile Tab:** This display user's profile information such as name, department, email etc. These information can also be updated from same page.
    - **Office Hour/Schedule Tab:** This tab is for professors/advisors to add their office/appointment hours. Also, office location and appointment duration is added on this page.

- **Appointment Tab:** This tab displays all recently booked appointment or cancelled appointments. It also provides the option to cancel an appointment. Once an appointment is cancelled by a professor/advisor, a notification is sent to the student who scheduled the appointment.

RESTFul SERVICE API:

I used slim framework to design my Restful Service API. Slim framework is an open source PHP micro framework which has the basic components such as URL routing, request and response services needed to build a good API. The main page where all URL routing is done is in “index.php”. This page contains all URL used to make all API calls required in Meliora Scheduler.

Just like any other Restful Service API, this API returns a “json” string as response. The response has two keys; an “error” key and a “message” key. If an error occurred during API call, the “error” key value will be “true” otherwise it will be “false”. The “message” contains the requested information from API otherwise it will contain error message.

Apart from “Sign up” and “login”, every other API calls requires an API key used to authenticate the API caller. A unique API key is generated by during signup process and stores in database