

Ketchup Clinic Full Design

I. Overview

Lively is a platform that targets newly diagnosed Type 2 Diabetes patients to help them learn more about their pills and create an optimized drug schedule that factors in their eating, sleeping, or exercise habits. Type 2 Diabetes is a chronic illness that can involve taking many pills and this can be overwhelming to manage, especially for those who are not used to taking pills. Lively makes this transition smoother and easier to accept because it tells you the best possible time to take your medications based on your existing life. This is important because this allows users to continue living their normal lives and manage their condition without major disruptions. Existing solutions, such as Mayo Clinic, offer similar features in terms of educating users on the pill and its side-effects, but do little to create a personalized schedule.

II. Conceptual Design

Problem

Type 2 Diabetes is one of the most prevalent diseases in the U.S. The American Diabetes Association shows that “in 2015, 30.3 million Americans, or 9.4% of the population, had diabetes” and every year about “1.5 million Americans are diagnosed with diabetes” [1]. This chronic illness involves substantial life changes that can be overwhelming. Newly diagnosed patients struggle with managing the oral medications, understanding the side-effects, and creating a drug schedule that integrates with their current eating, sleeping, or exercising habits. Ketchup Clinic’s goal to build a platform that allows users to learn more about each medication and create an optimized drug schedule that factors in their eating, sleeping, or exercise habits.

[1]: <http://www.diabetes.org/diabetes-basics/statistics/>

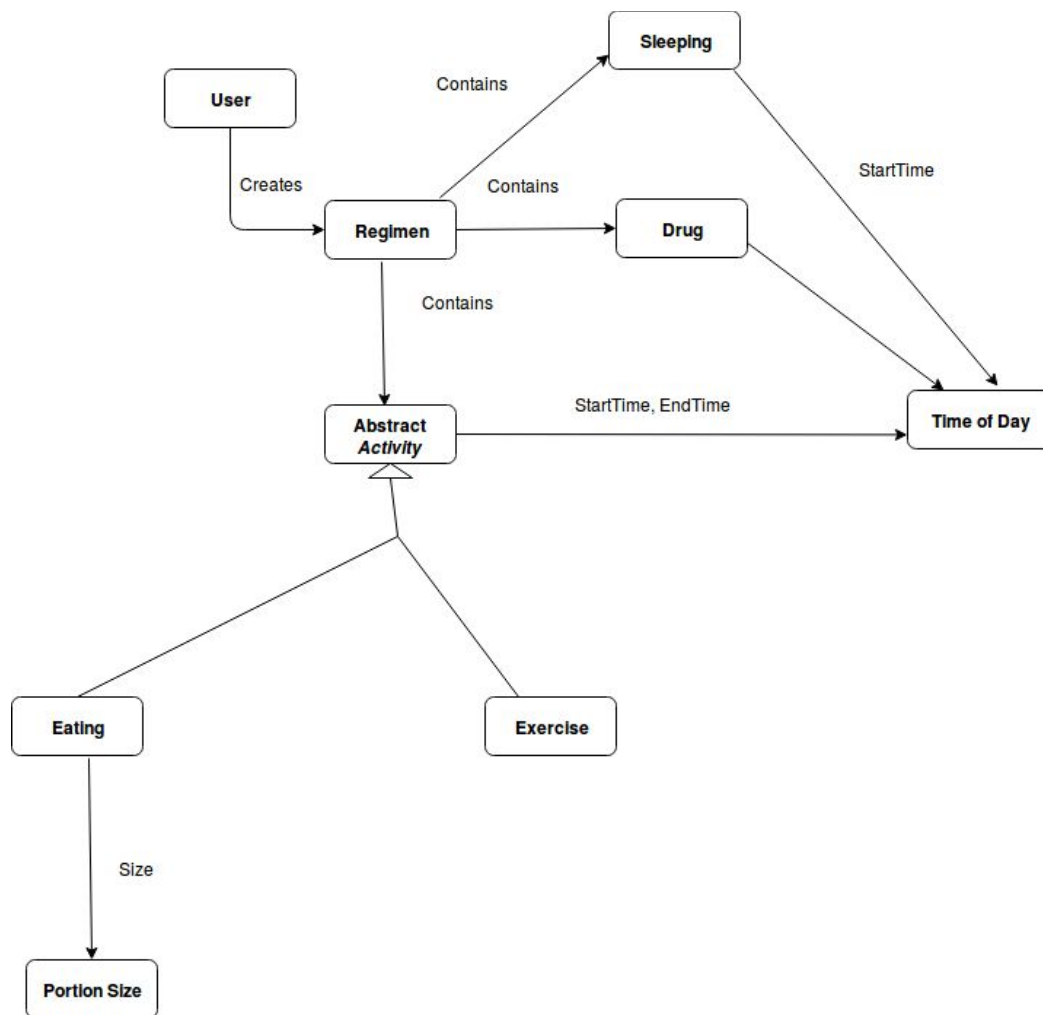
Project Key Concepts

I. Regimen

Purpose:

Allows a user to create a personalized drug schedule based on their sleeping, eating, and exercising constraints.

Abstract Data Model:



Actions:

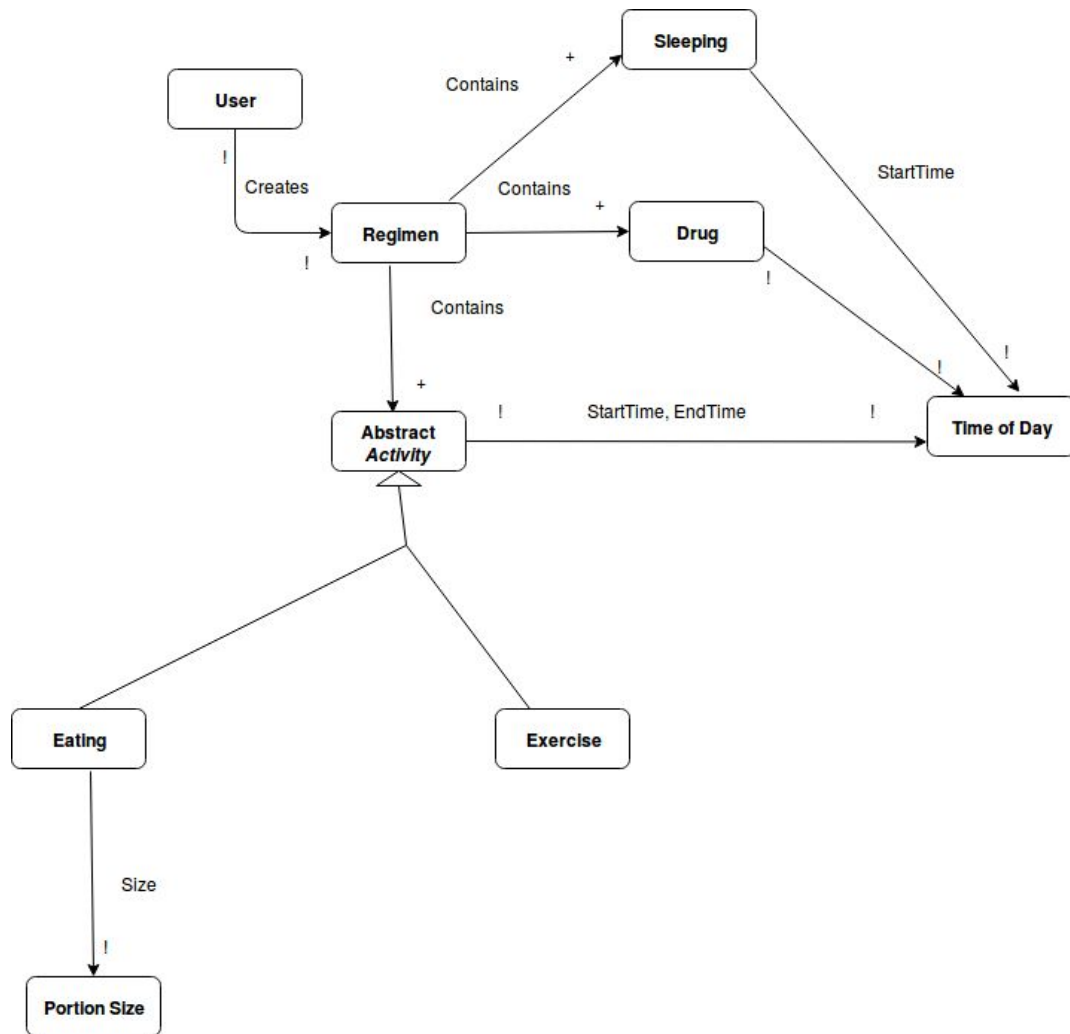
- **User: selectDrug()**
 - **Effect:** User selects from a list which drugs they were prescribed
- **User: addMeal(timeStart, timeEnd, mealSize)**
 - **Require:** mealSize can only be a value of: Light, Medium, Heavy
 - **Effect:** User creates a time period when they will eat and when they usually finish and how heavy the meal tends to be.
- **User: addTimeAsleep(timeStart, timeEnd)**
 - **Require:** timeStart != timeEnd

- Effect: User creates a time period where they will be asleep
- User: createRegimen(drugs, meals, asleep)
 - Require: drugs, meals, time asleep
 - Effect: User creates a drug schedule that integrates into their life

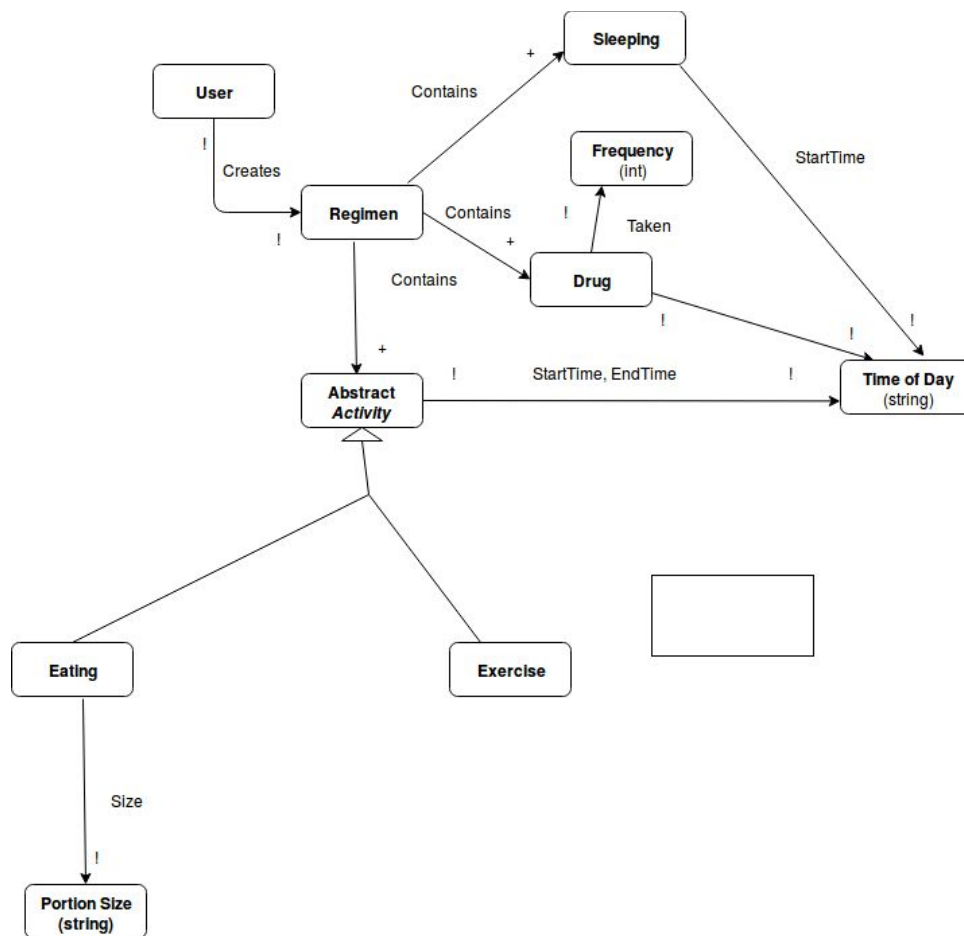
Scenario:

Joe Appleseed was recently discharged from Conceptual State Hospital with a diagnosis of Type 2 Diabetes. Joe received five pills he must take every day and was told to exercise five times a week to maintain a healthy blood sugar. While Joe received instructions with each pill he is scared and intimidated. Joe learns about Sugarly, a platform made by the Ketchup Clinic that helps people understand Type 2 Diabetes medications and create a manageable schedule. Joe signs up and selects the drugs he was prescribed. The system then asks Joe to add when he likes to eat and sleep. Joe then adds when would be a good time to exercise for 30 minutes. Finally, Joe clicks "Create Regimen" and a schedule for him is created that he now can follow everyday.

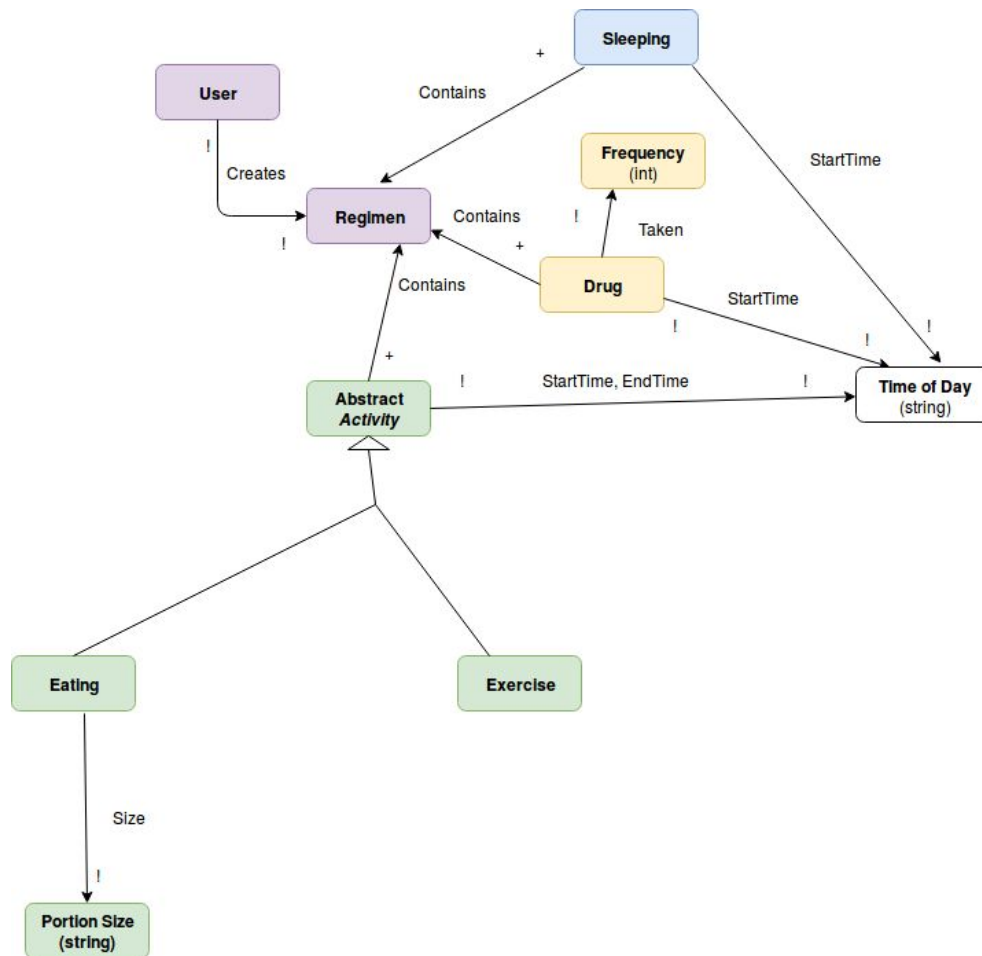
III. Data Model



Adding Primitives:



Adding Trees:



Constraints:

- We limit meal size to three categories: light, medium, and heavy to simple measurement of a meal
- An Activity can be associated to Eating, Sleeping, Exercise, or Occupation, but only one of them.
- StartTime and EndTime cannot be the same

IV. Schema Design

```
CREATE TABLE users {
    id VARCHAR(255) primary key not null unique,
    username VARCHAR(255) not null unique,
    password VARCHAR(255) not null,
    onboardingComplete bool(1)
```

```
}
```

```
CREATE TABLE drugs{  
    id VARCHAR(255) primary key not null unique,  
    name VARCHAR(255) not null unique,  
    frequency INT not null,  
    time_of_day VARCHAR(255) not null,  
    description VARCHAR(255) primary key not null unique
```

```
}
```

```
CREATE TABLE drugsRegimen{  
    userId VARCHAR(255) REFERENCES regimen(id),  
    drugId VARCHAR(255) REFERENCES drugs(id),  
    startTime VARCHAR(255) not null,  
    endTime VARCHAR(255) not null,
```

```
}
```

```
CREATE TABLE meals{  
    id VARCHAR(255) primary key not null unique,  
    name VARCHAR(255) not null unique,  
    size VARCHAR(255) not null,
```

```
}
```

```
CREATE TABLE mealsRegimen{  
    regimenId VARCHAR(255) REFERENCES regimen(id),  
    mealId VARCHAR(255) REFERENCES meals(id),  
    startTime VARCHAR(255) not null,  
    endTime VARCHAR(255) not null,  
    dayOfWeek VARCHAR(255) not null,
```

```
}
```

```
CREATE TABLE sleepSchedule{  
    startTime VARCHAR(255) not null,  
    dayOfWeek VARCHAR(255) not null,  
    userId VARCHAR(255) REFERENCES regimen(id),
```

```
}
```

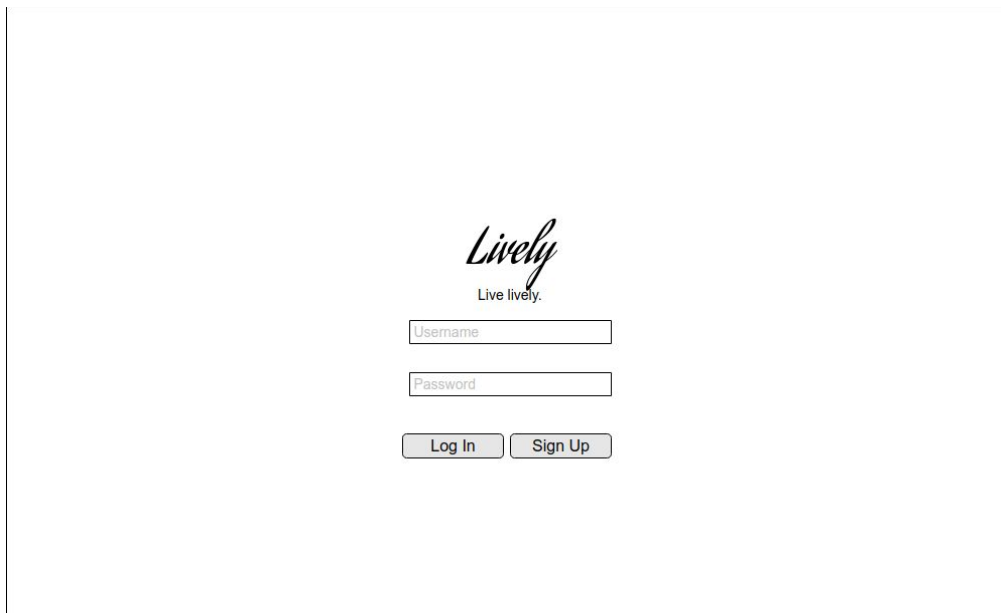
```
CREATE TABLE exerciseSchedule{  
    id VARCHAR(255) primary key not null unique,  
    name VARCHAR(255) not null,  
    startTime VARCHAR(255) not null,  
    endTime VARCHAR(255) not null,  
    dayOfWeek VARCHAR(255) not null,  
    userId VARCHAR(255) REFERENCES regimen(id),
```

```
}
```

V. Security Concerns

- Protecting personal information, especially sensitive health information such as what drugs a person takes
- SQL injections based attacks would be risk due to sensitive data being leaked
- Sanitize inputs because a lot of user input data

VI. Wireframes



The wireframe shows a login page for a service called 'Lively'. At the top center is the 'Lively' logo in a script font, with the tagline 'Live lively.' underneath it. Below the logo are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. At the bottom of the form are two buttons: 'Log In' and 'Sign Up'.

This is the first page a new user would see, a textbook example of a login page.

Select Drug	▼
Drug 4	+
Drug 5	+
Drug 6	+

Drug 1

Info

X

Drug 2

Info

X

Drug 3

Info

X

Next

Every user has a schedule already created, so new users are guided through the steps of making one. Here they select drugs from a drop down menu above and they are added to the collection below. They may undo selections by clicking the x buttons.

Activities

Meals

Activity Name

Start Time

End Time

Meal Size

☐ S ☐ M ☐ T ☐ W

☐ Th ☐ F ☐ Sat

Exercise

Activity Name

Start Time

End Time

☐ S ☐ M ☐ T ☐ W

☐ Th ☐ F ☐ Sat

Job

Activity Name

Start Time

End Time

☐ S ☐ M ☐ T ☐ W

☐ Th ☐ F ☐ Sat

Sleep

Activity Name

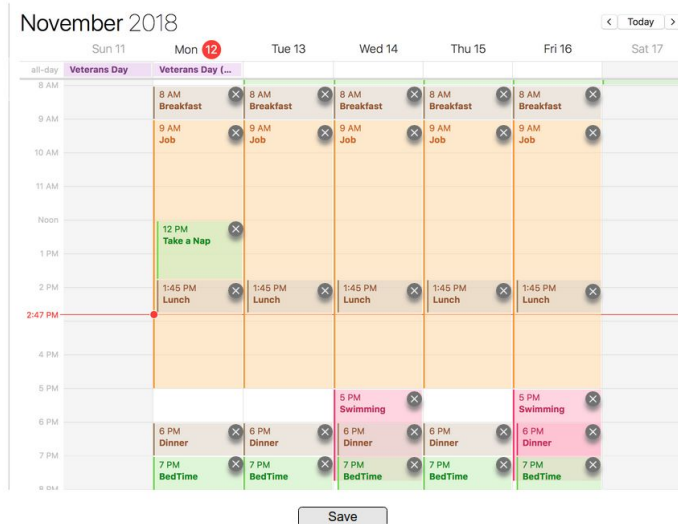
Start Time

End Time

☐ S ☐ M ☐ T ☐ W

☐ Th ☐ F ☐ Sat

Current Schedule

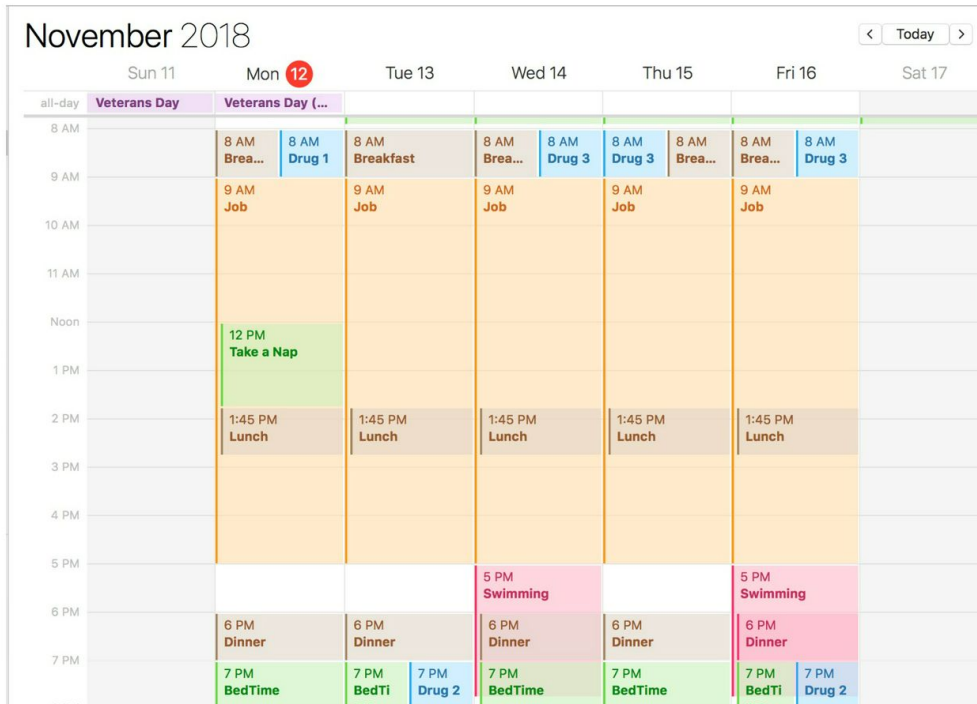


Here the user can then define four different types of activities they may partake in throughout their week as well as their times. This populates their schedule in realtime and gives something for our algorithm to work with. Once they save, the algorithm runs.

Current Schedule

Edit Activities

Edit Medication



When the algorithm finishes, the user is brought to their homepage, which is mostly just their calendar with newly added suggestions of when they should take their medications. They have the option to go back and edit some of their activities and drugs to update their schedule.

VII. Design Commentary

- We decided everything should have a start and end time in order to have a standard unit for all constraints
- We will store time in military time to avoid complications of storing am and pm time.
- We limit meal size to three categories: light, medium, and heavy to simple measurement of a meal
- We require at least one exercise activity because it is part of the treatment for Type 2 Diabetes
- The mealsRegimen, sleepSchedule, drugsRegimen, exerciseSchedule data tables reference a regimenId in order to correctly identify which schedule the activity belongs to

and subsequently which user the activity belongs to. Here we we have a design that favors correctness instead of one that favors efficiency.

- We will add token checking to deal with CSRF attacks

Algorithm

The algorithm attempts to place drugs in open slots during the day, but it follows a few rules:

1. Drugs can only be assigned during the hours after the time the user states they will be awake.
2. Drugs are preferably taken after the first meal.
3. Drugs are preferably taken right after every meal.
4. Drugs will not be taken the half hour before exercise is scheduled

If there is no way to satisfy any of these conditions the drugs will be scheduled as early as possibly, otherwise the user will be notified.

The algorithm that creates the representation has a few rules as well.

1. Each day is an array in an object.
2. The array only contains half hour blocks starting from the time when the user is awake.
3. The day currently ends at around 11:00pm