# Preliminary Design

## Primary Authors

- **Everyone:** data model, user interface
- **Jon:** key concepts, feature descriptions, tasks
- **Michelle:** security concerns, stakeholders
- **Tyler:** purpose and goals, design challenges
- **Rebecca:** context diagram, resources, risks, MVP

## Overview

### Purpose and Goals:

#### System To Be Built - Fireplace

Fireplace looks very much like a standard todo list app, but for any given state of the user (time, day, location) only tasks relevant to this state are displayed. Upon creating a task, the user provides a small amount of metadata that the app uses each time the user views his/her Fireplace to decide whether or not to display that task. These display decisions happen automatically without the user having to specify anything at the time of viewing, unless they want to utilize a few extra controls Fireplace will provide like "See All". The user can also package common sets of this metadata (sections of the day/week or locations like home/work they often want to partition certain tasks under) within something called a Tag, easing the process of applying this metadata upon task creation, so the user can get short accurate task lists with minimal extra work.

#### Key Goals And Purpose

The issue with showing all tasks every time the user views his/her list is that many of these tasks are impossible or irrelevant for the current state of the user. For example, people don't always want to see work todos when they are relaxing at home at the end of the day. Seeing everything is both inefficient (have to scan through a large number of tasks) and overwhelming (seeing so many things is stressful). Limiting the number of tasks based on relevance to the user's current state shortens the list of tasks shown to the user without hiding things they need to see at that moment, thus increasing efficiency and decreasing stress.

#### Motivation For Development

Fireplace is not trying to revolutionize or shift the current todo list paradigm. It actually mimics current solutions very much, but with the key difference of automatically filtering which tasks are shown based on the current state of the user. It is simply refining these current solutions since they have the issue of showing the user too much too often.

### Context Diagram:

Refer to 'figures/context_diagram.png'

## Concepts

### Key Concepts

- **Task:** A piece of work that the user wishes to complete.
- **Flag:** A boolean attribute associated with a task that influences when the task is shown. Fireplace will have flags for marking a task as important and/or long-lasting.
- **Metadatum:** A piece of datum associated with a task that allows the app to filter tasks. Three categories of metadata: times, days, and locations.
- **Tag:** Commonly used metadata that are grouped together and given a name.
- **Filtering policy:** A set of rules governing which tasks to show based on a time, date, and

location.

### Data Model

Refer to 'figures/data_model.png'

# Behavior

### Feature Descriptions

- **Managing tasks:** Users can create tasks, as well as view/edit/delete tasks that they created. Users can neither share tasks with other users nor interact with other users' tasks. A task is represented by a short, single line descriptor and a longer plain text summary. The user can attach tags to a task or specify the task metadata through a simple form.
- **Create tags:** Users can create tags to simplify adding the same metadata to multiple tasks. Tags can inherit metadata from other tags and/or specify metadata that overrides the inherited metadata. Tags are private to the user that created them, and cannot be shared with other users.
- **View tasks:** Fireplace will detect the user's location and local time, and then display all of the user's tasks that are applicable. The user can customize the task filtering policy by selecting to show/hide long-lasting tasks, show only important tasks, and/or show all tasks for the day/next day/week (filtering policy will ignore location).

### Security Concerns:

#### Security Requirements

Fireplace has minimal security requirements:

Users only have read and write access to their To-Do lists and their personal information. Personal information includes addresses, their current location, and email address.

#### Potential Risks

Users and developers of Fireplace should be aware of potential risks associated when using Fireplace:

In the event of a breach in security, users must be aware that their addresses, current location, email and password could be compromised. Since Fireplace depends on this information and users cannot withhold this information to use this app, users should proceed use of Fireplace with caution.

#### Threat Model

Fireplace is meant for personal use and does not allow any interaction with other users. This decreases the point of entry in which a malicious user could gain access to sensitive information. Attackers could try injection attacks to gain access to all sensitive information, CSRF to gain access to another user's account, and XSS to trick users into giving the attacker their personal information, as a few examples. The only assumption we can make with our application is that an attacker cannot gain direct access to our database as a result of using Heroku to deploy Fireplace.

#### Mitigations

- The minimum password length we require is 6 characters in order to prevent unauthorized access to accounts.
- Passwords will be encrypted in the database to protect user accounts in the case of a security breach. We trust Heroku to keep our database secure from unauthorized access.
- Saved addresses will be encrypted in the database.
- We will implement user access control to prevent unauthorized access to user accounts.
- We will rely heavily on Rails ActiveRecord since they escape strings when dealing with queries to the database, effectively preventing SQL injections.
- Rails protects against CSRF by default by including protect_from_forgery with: :exception in each

controller.
- We will use SSL to protect against attackers performing man-in-the-middle attacks or reading our data.
- Rails also escapes strings rendered in views to prevent XSS attacks.
- We will always use ActionController::Parameters require and permit to restrict what parameters we accept in the controller, which also protects against XSS attacks.

### User Interface

See figures folder for workflow wireframe and individual page wireframes

## Challenges

### Design Challenges

- Number of Tags per Task - How should the user be encouraged to think of tags?
    - Allow At Most 1 Tag per Task

      pushes the idea that Tags are disjoint categories into which a Task can be classified Pro: shortens process of choosing Tags upon Task creation since once a user chooses 1 Tag, they are done Con: means that if a user wants multiple tags to apply to a Task, they must first create a composite Tag, then apply that Tag to the Task in question
    - Allow Multiple Tags per Task

      pushes the idea that Tags are like Piazza tags or Twitter hashtags (hence calling them "Tags")
      Pro: more user control when creating Tasks Con: possibility of complicating the Task creation process
    - Proposed Solution: Allow Multiple Tags per Task
- Overriding Automatic Hiding of Tasks - How should the user be able to ensure they do not miss relevant Tasks?
    - Include Randomness In Display Condition

      Pro or Con: any Task theoretically could be shown at any time Con: does not guarantee problem is solved, since app could get extremely unlucky and still hide things the user wants
    - "See All Tasks" button

      Pro: blanket solution, solves all problems regarding hiding too much Con: if used too often, defeats purpose of Fireplace Pro: possible variation on this solution is have incremental stages in between fully automatic filtering and seeing everything, for example "See Today Tasks" which ignores time and location
    - Proposed Solution: "See All Tasks" button (or some variation)
- Non-periodic Conditions - What about Tasks whose relevance depends not only on day of week and time of day but also on something non-periodic, like a due date?
    - Allow Due Date Input

      Con: more input means more work for user Con: user would also need to specify how long before the due date they want to start seeing the Task Pro: grants user commonly desired feature of hiding Tasks that are not due in the near future
    - Only Allow Day of Week, Time of Day, Location

Pro: do not have to worry about details of due dates Con: ignores popular trend that users value a Task more highly (the Task become more relevant) as the due date approaches

- Proposed Solution: Allow Due Date Input
- Number of Different Pages - Across how many pages should all the features (viewing, creating, editing) be?
    - Single Page For Everything

      Pro: dashboard feel, like everything you need is easily accessible Pro: managing Tasks and Tags do not feel like arduous tasks that require entire new pages to do Con: must be careful not to clutter the page since space is limited so everything would be in close proximity

    - Separate Pages For Separate Functionalities

      Con: seems more difficult to perform simple tasks like creating or editing if a new page is rendered and the view of the to-do list itself disappears

    - Proposed Solution: Single Page For Everything

      Pro: user can be easily guided through the process of performing the different functionalities Fireplace provides, with little variation and thus less room for error and confusion by the user