MAST - My Advising Scheduling Tool Requirements

Target release	1.0
Document status	Final
Document owner	CS419-400 Group 1
Design Head	Bryon Burleigh
Development Head	Freelin Hummel
QA Head	Kabir Kang

1. Introduction

1.1. This document is meant to define the product we are creating as well as the requirements that come from its features. It begins by providing descriptions of the product, such as what it does and who it is for. It then breaks down the functionality of MAST (My Advising Scheduling Tool). The features that are described are illustrated by use cases and diagrams. This document will answer the following questions and more:

- **1.1.1.** What happens when an appointment is created for an advisor?
- **1.1.2.** What are the non-functional requirements of MAST?
- **1.1.3.** Can someone sign up for slots that are already occupied?
- **1.1.4.** Does MAST ever query an advisor's calendar?
- **1.1.5.** What are the database requirements of this tool?

2. General Description

2.1. **Product Description:** MAST will be a software tool designed to streamline advising appointments. It will be an integral piece of the academic advising process for any college or university that chooses to implement it. By automating the tedious task of processing advising meeting requests and cancellations, academic advising professionals can spend more time directly interacting with students throughout the academic year. This will be especially useful during busy course registration periods. Additionally, the time saved on processing meeting requests will allow advisors to efficiently assist a higher number of students in the same amount of time.

2.2. **Product Functions:** The primary function of this tool is to automate the creation and cancellation of student academic advising appointments. There will be two main components to MAST.

2.2.1. The first part consists of a tool that will parse emails created by the appointment system. It will use these emails to create appointment requests or cancellations on an advisor's Outlook or Gmail calendar.

2.2.2. The second part is a CLI client (command line interface) that will allow an advisor to manage their appointments from a terminal and affect the calendar appropriately.

2.3. Users: The CLI client and procmail filter are purposed for student advisors. Advisors will have the ability to affect their own calendars with the command line tool. Emails created by the advisement appointment website will be sent to these users and generate requests or cancellations appropriately.

2.3.1. Note: Students will only interact with the existing advising web form to request or cancel a meeting with their academic advisor. Student interaction is not within the scope of this project.

2.4. Assumptions

2.4.1. Webform for students to pick available advising times already exists, and it will generate the emails that MAST will process to schedule advisement meetings.

2.4.2. Existing webform only allows students to schedule available times. There is no need for MAST to query advisor calendars for available meeting times.

2.4.3. Client will provide database to use, and the development team will have the ability to manipulate tables as needed or be provided with a schema for any existing tables the client wants to be used

2.4.4. Meeting requests and cancellations must be readable by the email client.

3. System Requirements

3.1. General Requirements

3.1.1. Procmail Filter: Converts emails from text to a meeting request or cancellation. It also updates the database which contains the current appointments.

3.1.2. Command Line Client: This allows command line management of the database. Appointments must be viewable and cancellable in this interface. Cancelling affects the calendar.

3.2. Functional Requirements

3.2.1. Procmail Filter

3.2.1.1. Overview: If an email is a meeting cancellation or request, appointment information is converted to standard Outlook meeting request format and affects the calendar appropriately. This information will be viewable in the CLI client.

3.2.1.2. Inputs: Plain text emails from EECS advising web form

3.2.1.3. **Outputs:** Meeting request to the advisor's calendar and appointment information for the mysql database

3.2.2. CLI Client

3.2.2.1. **Overview:** Every appointment an advisor has in the database can be viewed via the MAST command line UI. Appointments can be cancelled from the advisor's calendar and removed from the database via the MAST client. This does not remove the appointment from the advising system.

3.2.2.2. **Inputs:** Appointment data from the MySQL database

3.2.2.3. **Outputs:** Appointment data to screen; a cancellation email and calendar removal in the event of an appointment cancellation

3.3. Use Cases From Client

3.3.1. Student Cancels Appointment

3.3.1.1. Kevin, a student, has already scheduled an appointment, and decides to cancel it. This prompts the system to send an email to his advisor. This email contains a meeting cancellation, and so the advisors calendar client removes the appointment from his calendar.

3.3.2. Advisor Cancels Appointment via Client

3.3.2.1. An advisor prefers to use Linux computers, and is making use of the command line client. He realizes a student who is not his has signed up for an appointment, and cancels this appointment. While this doesn't remove it from the advising system, it does remove it from the advisors calendar via a meeting cancellation email.

3.3.3. Student Creates Appointment

3.3.3.1. A student signs up for an appointment, and the email received by the advisor contains a meeting request, which adds the student to the calendar.

3.4. User Tasks

3.4.1. Student

3.4.1.1. Creates or cancels appointments through the EECS advising web form

3.4.1.2. For each appointment created or cancelled, the form sends the advisor an email.

3.4.2. Advisor

3.4.2.1. The procmail filter creates and cancels appointments from EECS advisement emails.

3.4.2.2. With the CLI client, the advisor can view and delete appointments from the calendar.

3.5. Non-Functional Requirements

3.5.1. **Reliability:** A bug in this system could compromise meetings between students and advisors. If calendars are not affected correctly, the system would be of no use to its users. In addition, the CLI client must adhere to usability standards so that advisors can make proper use of it.

3.5.2. **Availability:** The ability to view and manage appointments via the CLI client must be possible on a variety of systems, from an advisor's laptop to a department LINUX machine.

3.5.3. **Security:** To avoid compromising the actual advisement schedule, the database and client shall only be for the purposes of advisor calendars. The MySQL database must be safe from injection or deletion of entries from foreign parties.

3.6. Design Constraints

3.6.1. The client database will be a mysql database.

3.6.2. The procmail filter and MAST client will be written in Python.

3.6.3. MAST client will be a curses-based CLI.

3.7. Inverse Requirements

3.7.1. Development team will not be implementing any interface with existing academic advising infrastructure beyond processing emails and creating/cancelling meeting appointments on individual advisor calendars.

3.7.2. Cancellations via the CLI client will not generate emails for students. This action will remove the appointment from the advisor's calendar only.

4. Flow Diagram