Product Backlog

Guna Kondapaneni, Lisa Campbell, Matthew Pace,
Brian Kaetzel, Joseph Khawly

Problem Statement:

Everyone faces a very stressful workload and needs a quick and practical way to manage their busy, sometimes hectic, lives. We will provide a smart calendar that helps them recognize and manage the chaos of life. The calendar will integrate with the user's existing calendars, determine which events are stressful, provide these details to the user, and recommend ways the user can de-stress their life. For example, if meetings stress a particular user, and on Tuesday, they have a day full of meetings but there are none on Wednesday. The app will suggest the user to move one of the meetings to the next day, if possible, and it will pick the meeting with the least number of people included. If moving a stressful event is not possible, it will instead suggest adding a de-stressing activity to the user's calendar.

Background Information:

Stress has become even more of a problem than it has in the past. According to the American Psychological Association, "Adults rate their average stress level as 5.1 on a 10-point scale, where 1 is "little or no stress" and 10 is "a great deal of stress," slightly up from 4.9 in 2014." This means that stress levels have gone up in adults even in a year's time. This demonstrates a need for better stress management. Our app's goal is to help with effective stress management and to help people realize when their stress levels are getting too high.

Environment:

For the front-end, it will be in ReactJS and the back-end will be in Java, using the Spring framework. This app will use Google Calendar's external API's to access and manipulate a user's calendar. Our database will be Amazon's DynamoDB. User sign in will be handled with Google Sign-in.

Functional Requirements:

Backlog Id	**Functional Requirement**	**Hours**	**Status**
1	As a new user, I would like to be able to create a new user. (only Google OAuth login, includes setup)	~~12~~ 20	Completed sprint 1
2	As an existing user, I would like to be able to log into the	~~3~~ 5	Completed sprint 1

	app. (only Google OAuth Login)		
3	As an existing user, I would like to be able to import my Google Calendar.	~~6~~ 10	In-progress: moved to sprint 2
4	As an existing user, I would like to be able to rate how stressful or de-stressing certain events are on my calendar.	12	Incomplete: moved to sprint 2
5	As an existing user, I would like to see how stressful each day in my upcoming week will be.	14	Planned for sprint 2
6	As an existing user, I would like to be able to see suggestions popup to help me make my extremely stressful days less stressful. (aka. modify schedule)	8	Planned for sprint 2
7	As an existing user, I would like to be able to synchronize my calendars.	5	Planned for sprint 2
8	As an existing user, I would like to be able to manually set my stress level of an event	3	Planned for sprint 2
	Total:	~~63~~ 77	

Non-Functional Requirements:

User registration and login will use OAuth 2.0 protocol for authentication and authorization. The database chosen is a scalable, NoSQL database called DynamoDB. The app will be hosted through Heroku for easy recovery of logs in case of errors.

Use Cases:

Case: New User Creating An Account

Action	System Response
1.) Clicks register button 3a.) Enters in all necessary, unique, user information in form and clicks register 3b.) Enters user information already in use by an existing user 3c.) Enters incompatible user information	2.) Navigates to registration form 4a.) Stores all user data and logs them in to find a stresses form 4b.) Reports an error to the user 4c.) Reports an error to the user

Case: Existing User Logs In

Action	System Response
1.) Clicks login button 3a.) Enters valid username and password 3b.) Enters invalid username and password	2.) Navigates to login form 4a.) Validates that login information is correct and if so, logs them in 4b.) Displays error and asks user to try again

Case: Import Google Calendar

Action	System Response
1.) Logged in user clicks on import calendar button 3.) User is prompted to rate stress of imported events (see stress form case)	2.) Uses Google's API to retrieve the user's Google calendar and imports it into the stress calendar

Case: Rate Stressfulness of Events Manually

Action	System Response
1.) User clicks on rate event button 3.) User selects the stressfulness of event	2.) Dialogue box opens with various stress ratings 4.) All events with the same name are stored with selected stressfulness level

Case: View Stress Level of Week

Action	System Response
1.) User clicks view weekly stress button 3.) User clicks close in the stress window	2.) Stress of all events for the week are added and displays information in a window 4.) Window closes

Case: View Recommended De-Stressed Week

Action	System Response
1.) User selects de-stress my week button 3.) User clicks accept or decline for each event recommendation	2.) Optimum week is calculated and displayed in a new window 4.) Old schedule is replaced with de-stressed calendar

Case: Stress Form

Action	System Response
1.) User inputs valid input (integers from -10 to 10 in the form via a slider) for stressor amounts 3.) User clicks submit	2.) Each event is tagged with the stress level the user inputted

Case: Synchronize Calendars

Action	System Response
1.) User allows write access during registration	2.) Events are bi-directionally passed between stress calendar and Google calendar continuously