ManageHelp

Sprint 1 Retrospective

Team 24

Tom Carsello
Matt Hiatt
Sharan Sivakumar
Jon Tokad

What went well during Sprint 1?

After a lot of research, practicing, and prototyping, we as a group are now well versed enough to complete user stories without having to spend absurd amounts of time and effort to make sure we are using the MERN stack to its full capabilities. The skeleton of the entire website has been created, and now we aren't dependent on a single person's contribution to continue to make changes to the project. As a team, we found out that we work best when collaborating in a single controlled environment at consistent times, rather than working alone and merely stating what we've achieved on our own. We were able to help each other out and share collaboration on most of our tasks.

Completed

User Story (1/12): *As a user, I would like to be able to create and log in to my ManageHelp account*

Task #	Description	Time	Owner
1	Configure authentication	5h	Matt
2	Pipe user information to database	1h	Matt
3	Implement UI for account creation and login	4h	Sharan
4	Implement login credential validation and route user to correct page	3h	Matt
5	Use unit tests to debug and verify the authentication system works	4h	Jon

Acceptance Criteria

- Given that the user's email is not already associated with an account and that their password is not too weak, the user's account will be created and logged into.
- Given that the backend user login credential validation functionality works properly, a user should be able to input a username and password and the system should be logged in from then on.
- Given that the backend user login credential validation functionality works properly, once the user's account details are verified, the system needs to take the user to the correct home page based on their level of access
- Given that the backend user login credential validation functionality works properly, the system needs to notify the user whenever they input a wrong combination of either username or password.

Comments:

Users can create accounts on the signup page. If the email already exists or if the password is too weak the box turns red and the error is displayed. The app uses jsonwebtokens to keep users logged in for a day even if they close the app (unless they logout obviously.) After logging in, the user is routed to their homepage where they can select workspaces to view. If the user puts in the wrong password the box turns red and notifies them, if they are logging in with an email that isn't associated with an account, the email box turns red and it suggests they try signing up. We did not get to unit testing.

User Story (2/12): *As an admin, I would like to be able to create and edit my Company's ManageHelp Workspace*

#	**Description**	**Time**	**Owner**
1	Upon workspace creation, add workspace to database and set the creating user as its admin	4h	Jon
2	Implement create workspace UI	6h	Jon
3	Allow admin to change the name and join code for their organization workspace	1h	Jon
4	Use unit tests to assess functionality of the workspace creation	4h	Jon

Acceptance Criteria

- Given that the backend workspace functionality is implemented properly, when a user tries to create a workspace, the workspace will be created and the creator will have admin privileges for the workspace.
- Given that the admin has created their account successfully, the home screen should contain an option to view the workplace information like join code, company name, and privileged users.
- Given that the admin can properly view workspace information, they should also have the ability to edit particular fields and those changes should be visible on the frontend.

Comments:

Any user can create a workspace on the homepage, then they are set to admin of the workspace and it is displayed on their homepage. They can view and edit the workspace info once they click the workspace on the homepage. When they make changes the view page is updated to reflect the new values of the fields.

User Story (3/12) : *As an employee, I would like to be able to join my company's ManageHelp workspace*

#	**Description**	**Time**	**Owner**
1	Implement UI for user to input organizational join code	6h	Tom
2	Route user to the appropriate workspace	1h	Tom
3	Update workplace database with new user info	1h	Tom
4	Create unit tests to test employee joining ability	3h	Jon

<u>Acceptance Criteria</u>

- Given that a user has signed up for an employee account, they can then enter a workspace join code given to them by their boss (workplace admin.)
- Given that the user has supplied an invalid join code, the system must notify the user that the code was invalid.
- Given that a user has logged in, the system should list all workspaces they are a member of, and can click on them to access that workspace.
- Given that a user has joined a workspace as an employee, the user should see their role as "Employee" within the workspace

Comments:

On their homepage, users have the option to join a workspace by entering the workspace join code. If the join code is invalid, the error is displayed, the same occurs if they try to join the same workspace twice. If it is correct, the workspace is displayed on their homepage and they are added as an employee in the database. We did not get to unit testing.

User Story (4/12): *As a admin, I would like to be able to remove employees from the workplace*

#	**Description**	**Time**	**Owner**
1	Implement managerial removal function in backend	2h	Tom
2	Create remove tab and UI in web client	4h	Tom
3	Notify employee that they have been removed via email	2h	Tom

4	Unassociate user account with that workspace in the database	1h	Tom
5	Use unit tests to debug and verify removal function works	4h	Jon

Acceptance Criteria

- Given that a user is an admin for the workplace, they should be able to remove employees at their discretion.
- Given that a user has been removed from a workspace, the system should notify the user via email when the manager removes them from the workspace.
- Given that the functionality to remove employees is implemented correctly, the user data for that workplace should be wiped and they should only be able to access any other workspaces they were in

Comments:

The admin functions page contains an option to remove/fire employees. When an admin inputs the user to be deleted, they are wiped from the workspace database and sent an email notifying them that they have been removed from the appropriate workspace. We did not get to unit testing.

User Story (5/12): *As a admin, I would like to invite employees via email to join a workspace*

#	**Description**	**Time**	**Owner**
1	Set up mailer and ability to email users	7h	Matt
2	Implement email text with link to join an existing workplace	2h	Matt
3	Route email recipient to account creation page then to the appropriate workspace	1h	Matt
4	Use unit tests to test mailer works with dummy email addresses	4h	Matt

Acceptance Criteria

- Given that a workspace has been configured correctly, the admin should be able to send an email containing a link to join the workspace to a not-yet-registered employee.
- Given the recipient's email is not yet associated with a ManageHelp account, they will be guided to the account setup page.

- Given the recipient's email is already associated with a ManageHelp account, they will simply be added to the workspace once joining via the link.

Comments:

The mailer sends two different emails. If the email entered by the admin is already associated with a ManageHelp account, then they are sent a join link. If it is not, they are given the join code and sent a link to the signup page where they can make an account.

User Story (6/12): *As a manager, I would like to be able to assign job roles and pay rates to employees*

#	Description	Time	Owner
1	Implement managerial role assignment function	1h	Tom
2	Notify employees when there has been a change in their role or pay rate	3h	Tom
3	Implement managerial pay assignment function	1h	Tom
4	Implement admin role assignment function	1h	Tom
5	Test assignment functionality with written unit tests	2h	Tom

Acceptance Criteria

- Assuming the user is an admin account, they should be able to assign roles to various employees
- Assuming the user is a manager, they should be able to assign pay rates to various employees
- Assuming the user has signed up for a managerial account, the system should notify the user when there has been a change in their current roles/pay rates.

Comments:

Manager and the admin can both assign pay rates to the regular employees in their workspaces. The admin can also update the employee roles (ex. 'Cook' or 'Cashier'.) Those roles are displayed on the employee cards in the view function. The mailer sends a notice when users information is changed.

User Story (7/12): *As a manager, I would like to be able to view information about my employees in an easy to read format*

#	Description	Time	Owner
1	Retrieve all employee information from workspace database	2h	Tom
2	Implement UI to display all employees for manager to see, and enable their information to be shown if they are clicked on	4h	Tom
3	Allow employees to be listed or grouped by various attributes (pay rate, role, etc)	1h	Tom

Acceptance Criteria

- Given the manager selects the view employees button, they are presented with all of and only the employees in that workspace.
- When the manager views the employees, s/he should have the ability to change how they are listed/sorted.
- Given edit functionality is implemented properly, the manager should be easily able to move to the edit function from the view screen to adjust information

Comments:

The managers see the employees and their data listed on card-like objects that resemble the workspace cards that show on every user's homepage. The info is displayed when they select the dropdown for viewing employees on the admin function page and it is right next to the edit function for ease of access. We elected not to allow different sorting for now since there really isn't very many different fields that you want to display except for their name.

User Story (8/12): *As an employee, I would like to be able to input scheduling restrictions*

#	Description	Time	Owner
1	Implement UI for an employee to edit their scheduling restrictions	4h	Jon
2	Update scheduling data in the database	1h	Jon
3	Display restrictions for each employee when a manager is attempting to schedule them	4h	Jon

4	Write and test functionality with unit tests	4h	Matt

Acceptance Criteria

- Given that an employee has joined the workspace, they should be able to add times they can not work into their profile.
- Given that the input restrictions functionality is working properly, the manager should be able to see all employees restrictions in the employee view screen.
- Given that an employee has input restrictions, the scheduler should warn a manager when they are scheduling that person for their restricted times.

Comments:

Employees can input what days and times they can not regularly work, which is distinct from the day off request feature we did not get to. This way users do not have to repeatedly submit day off requests for recurring conflicts like school or other jobs.

User Story (9/12): *As an admin, I would like to be able to delegate scheduling permissions to shift managers*

#	**Description**	**Time**	**Owner**
1	Implement UI for an admit to delegate scheduling permissions to shift managers	4h	Sharan
2	Update schedule permissions in the database	1h	Sharan
3	Send email notifications when a shift manager's scheduling permissions are delegated	2h	Sharan

Acceptance Criteria

- Given that the user is an admin, they should be able to delegate scheduling permissions to shift managers
- Given that the user is an admin, when they delegate schedules to shift managers, the scheduling permissions should be accessible by the newly appointed managers.
- Given that the user is an admin, when they delegate schedules to shift managers, the shift managers should be notified by email

Comments:

The admin can delegate managers to be able to schedule employees from the admin functions page. The managers receive an email when this happens and then their account is updated to enable them to edit the schedules.

User Story (11/12): *As a user, I would like to be able to reset my password via (email, security questions?)*

#	**Description**	**Time**	**Owner**
1	Create UI for password recovery screen	3h	Sharan
2	Create password recovery system	3h	Sharan
3	Update database when a user's password is changed	1h	Sharan
4	Notify the user when their password is changed	1h	Sharan

Acceptance Criteria

- Given that a user has forgotten their password, any user should be able to reset their password via email
- Given that a user has begun the process of resetting their password, there should be clear instructions and a useful UI to guide them.
- Given that a user has changed their password, the system should notify the user of the change via email

Comments:

On the login screen, the user has an option to send a password reset email. They receive a randomly generated password that can be used to login, then they can change their password in the settings page. All passwords and salted and hashed before being stored in the database.

User Story (12/12): *As an admin, I would like to be able to promote existing employees to managers*

#	**Description**	**Time**	**Owner**
1	Create UI for admin being able to promote existing employees to managers	2h	Tom
2	Update the database when the permission of any user is changed	2h	Tom
3	Notify employees when their permissions are changed	1h	Tom

Acceptance Criterion

- Given that an admin has started to promote an existing employee, there should be a clear and intuitive user interface to guide them

- Given that a user is an admin, the user should be able to promote other users to the manager role
- Given that the user is an employee, notify them when their permissions are changed by admin

Comments:

We completed this user story without much problem, it was actually simpler than we expected. All it really ended up requiring was moving a user from one array to the other in the workspace schema and then their permissions were automatically updated because we had already set up all the routing to be based on the workspace arrays.

Not Completed

User Story (10/12): *As a user, I would like to be able to submit requests for days off to the manager for approval*

#	**Description**	**Time**	**Owner**
1	Create UI for a user to be able to submit requests for days of to the manager for approval	5h	Sharan
2	Update requests for days off in the database	5h	Sharan
3	Notify users when a manager approves their request for a day off	5h	Sharan

<u>Acceptance Criteria</u>

- The user should have a clean and intuitive user interface to be able to submit requests for days off to the manager for approval
- Given a request for a day off is made, the appropriate manager should be notified by email and able to approve it on their approval page
- When a user's day off is approved/rejected, they should be notified by email

Comments:

We still intend to implement this feature, but it was more of a technical challenge than anticipated and it really did not belong in this sprint. It proved very challenging to have the requests be approved by the appropriate personnel and updated properly. There are also other user stories that should be implemented first to provide the scheduling interface and give this user story a better location in the web application to be displayed.

What did not go well in Sprint 1?

The biggest issue in sprint 1 was our slow start, learning the stack was quite challenging as none of us had any real experience with javascript at all. In reality, we should have started learning with great urgency as soon as we knew the tech stack we were using, not waiting until the start of the sprint. Our communication also took a week to become as optimal as it should have been, there was a general fear in the beginning about not wanting to step on each other's toes but once we got over that we were able to be much more effective. However, we have improved greatly on that front, as outlined in the next section.

The biggest outcome of this mistake was the lack of time to implement user story 10. It would have been nice to get that out of the way as now we will have to do it in sprint 2 which adds another difficult user story to an already pretty tough sprint. We also did not get to the testing we had planned. We did all of our testing ourselves just inputting edge cases and writing the error handling when we found a new one. This approach is not efficient and we will have to learn proper testing for sprint 2 anyway as it is required.

How should we improve?

Over the course of the first sprint we made a lot of improvements in regards to communication and efficient workflow. In the second half/last week of the sprint we started meeting to do pretty much all of our work. This made it a lot easier to complete tasks since we could coordinate in real time on the different components of the user stories. In the beginning of the sprint we were mostly just divvying up the work based on the planning document and trying to set weekly goals for work to complete. This did not go so well as some people's user stories couldn't really be started until other people had finished a ton of their work for the sprint so it wasn't very balanced week to week. When we switched to working more closely together we were able to work faster and make the weekly work time for every member more equitable.

We also want to make more use of our weekly standup with Syed to ask more specific questions. In the first couple meetings we were more just using it as a check in but now that we have more expertise and direction we can ask for design advice, help with feature ideas, etc. We plan to maintain our end of sprint 1 approach going into sprint 2. If we can hit the ground running in the same form we were in for the last week of sprint 1 then sprint 2 and 3 will be a breeze.