SwIRL Final Report

Team Members:

Mohit Sarin
Erik Priest
Shaunak Joshi
Ishant Kundra
Aashay Kadakia
Chong Wen

Summary

The client wants us to create Skhedule.com, which will be a scheduling platform tailored for events that have specific attendance needs or capacities. In our proposed web application, there are two types of users (event organizer and attendee). As event organizers, users can create events with various options, such as making the event public or private, hosting it once or having it repeat, and adjusting the status of certain guests. As an attendee, users can choose which event to attend, select their preferred attendance time, book tickets, and cancel them at a later point. Users are required to create an account to send and view invitations but invitees are not. Instead, they can opt to receive an email confirmation with the details and have those details added to their calendar.

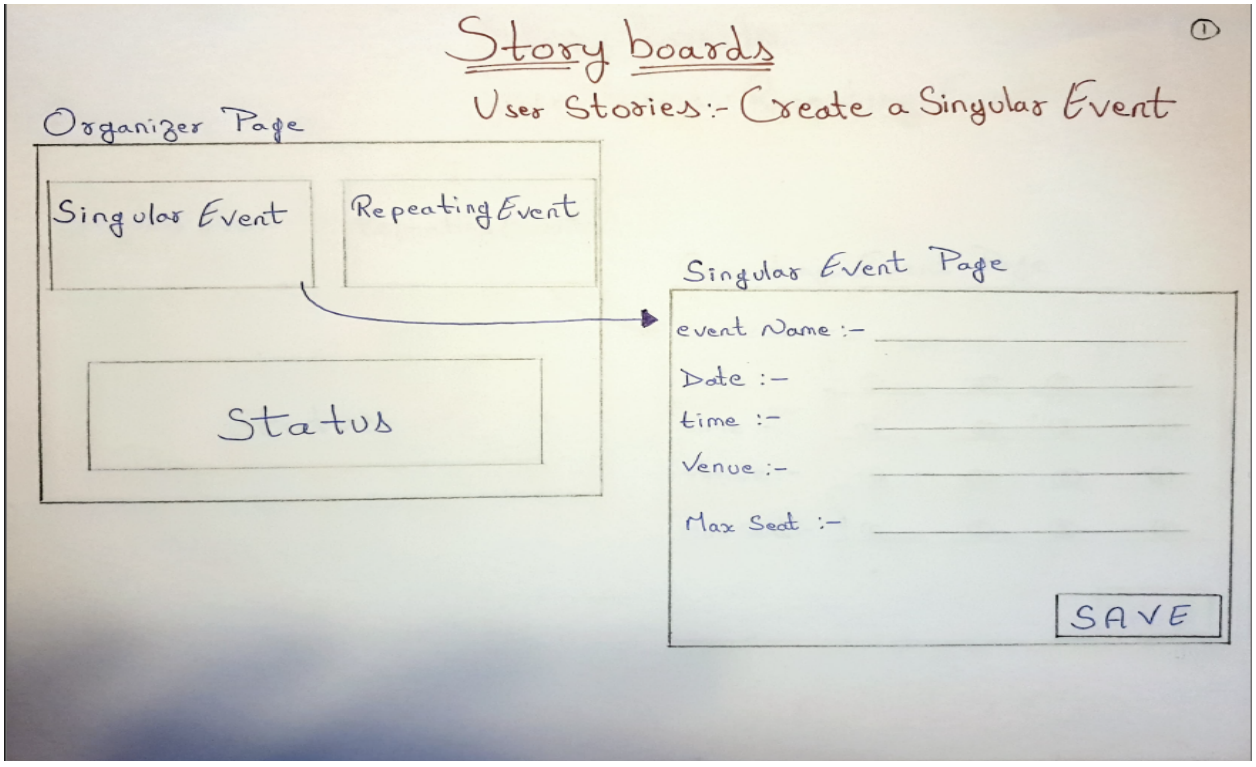
Further, upon logging in, the event coordinator or organizer is presented with an initial screen where they can choose between two types of events: "Singular" and "Series (Speaker)"
This singular type is designed for one-time occasions where a specific number of attendees need to be accommodated. The objective here is to invite individuals until the desired number of spots are filled. A practical example of this could be a dinner party, where the host aims to invite guests until all available seats are occupied. On the other hand, the Series (Speaker) event format is suitable for institutions like universities that wish to invite external experts to participate in a lecture series. Unlike one-time events, a Series involves multiple instances or sessions. It is similar to a speaker series, where various experts are invited to deliver talks on different occasions. This application perfectly meets the needs of the client. The Stakeholder of this web application is Southwest Innovation Research Lab (SwIRL) and they will use this application for event scheduling purposes.

User Stories

User Story 0: Create initial user stories, their Lo-Fi Mockups, and storyboards
Status: Implemented
Points: 32
Created initial user stories for iteration 0, their Lo-Fi Mockups, and storyboards that will help us in the future from the design point of view.

User Story 1: Create a Singular Event
Status: Implemented
Points: 8

As an event organizer, I want to create a one-time event on Skhedule.com efficiently. I need to specify the event's name, date, time, venue, and maximum seat capacity and optionally assign seating if required so that I can easily manage and organize unique events.

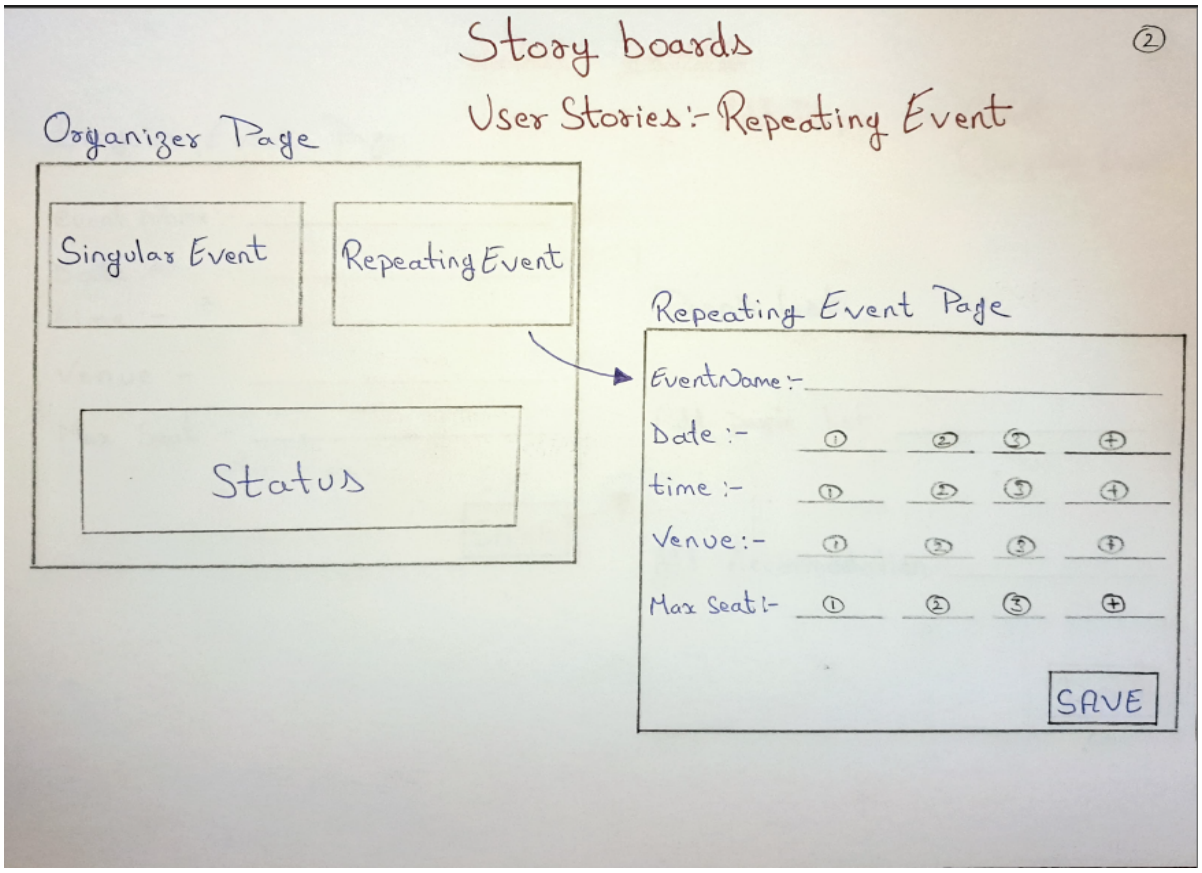


Create a Singular Event Storyboard

This Storyboard describes the way in which an organizer can create a singular event.

User Story 2: Create a Series Event
Status: Implemented
Points: 8

As an event organizer, I want to efficiently create recurring events on Skhedule.com. I need to set the event's name, date, time, venue, and maximum seat capacity, and assign seating if necessary, for each instance of the event, so that I can schedule and manage events that occur multiple times.

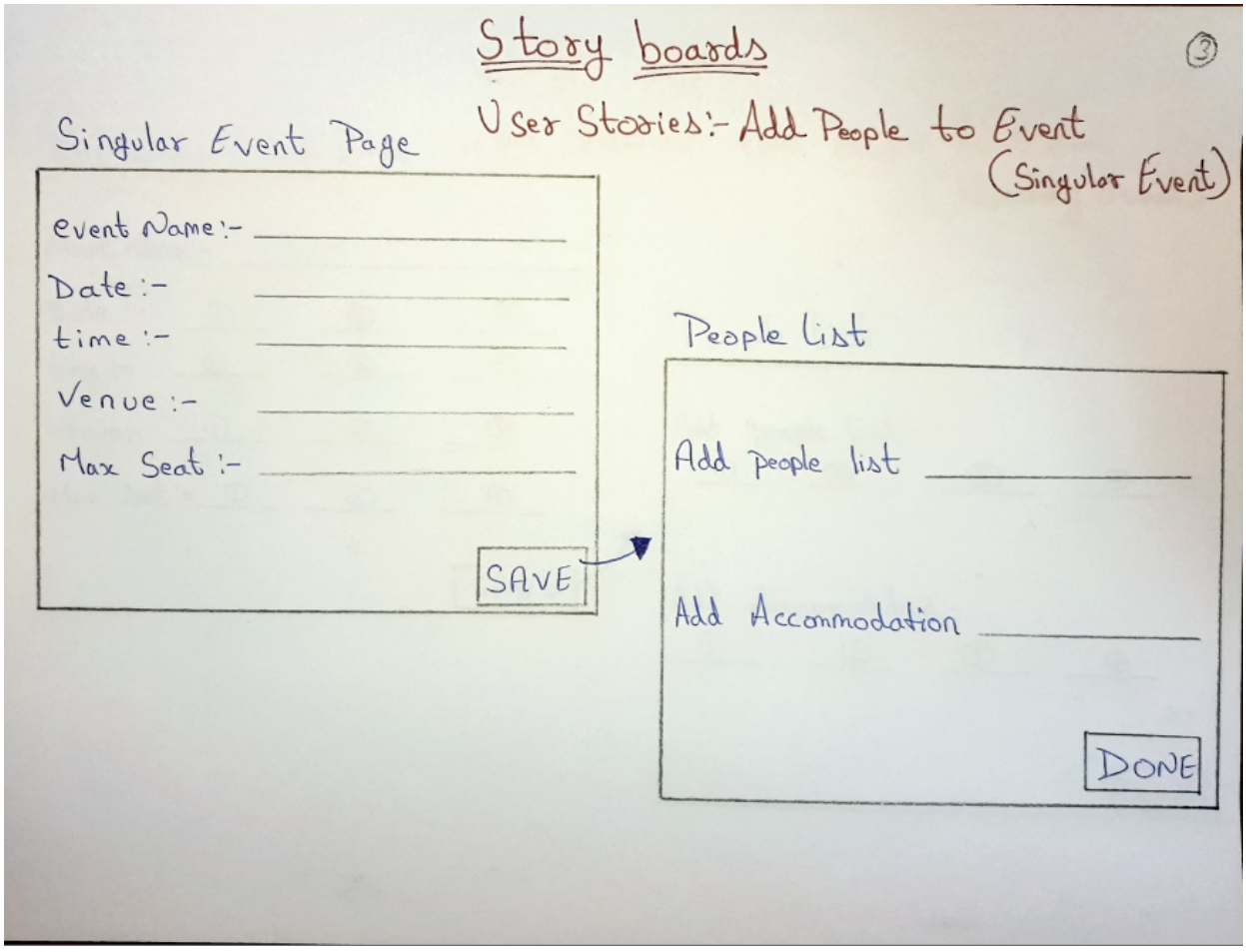


Create a Repeating Event Storyboard

This Storyboard describes the way in which an organizer can create a repeating event also called a series.

User Story 3: Add People to Event (Singular Event)
Status: Implemented
Points: 16

As an event organizer, I want to be able to add individuals to my events on skhedule.com. For each attendee, I need to record attributes such as dietary restrictions or disability accommodations, so that I can better accommodate the needs of each participant.

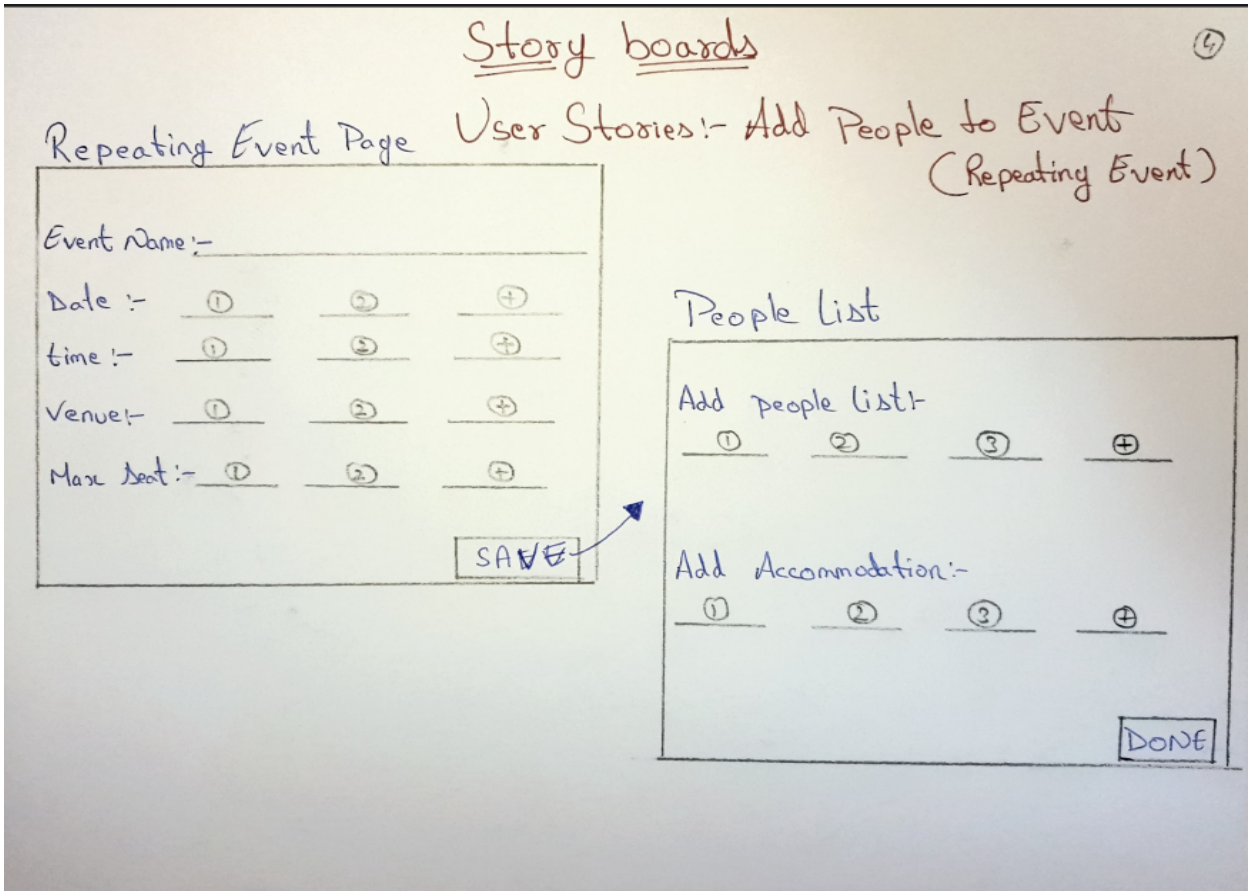


Add People to Event (Singular Event) Storyboard

This Storyboard describes the way in which an organizer can add people to a single event.

User Story 4: Add People to Event (Series Event)
Status: Implemented
Points: 16

As an event organizer, I want to be able to add individuals to my events on skhedule.com. For each attendee, I need to record attributes such as dietary restrictions or disability accommodations, so that I can better accommodate the needs of each participant.

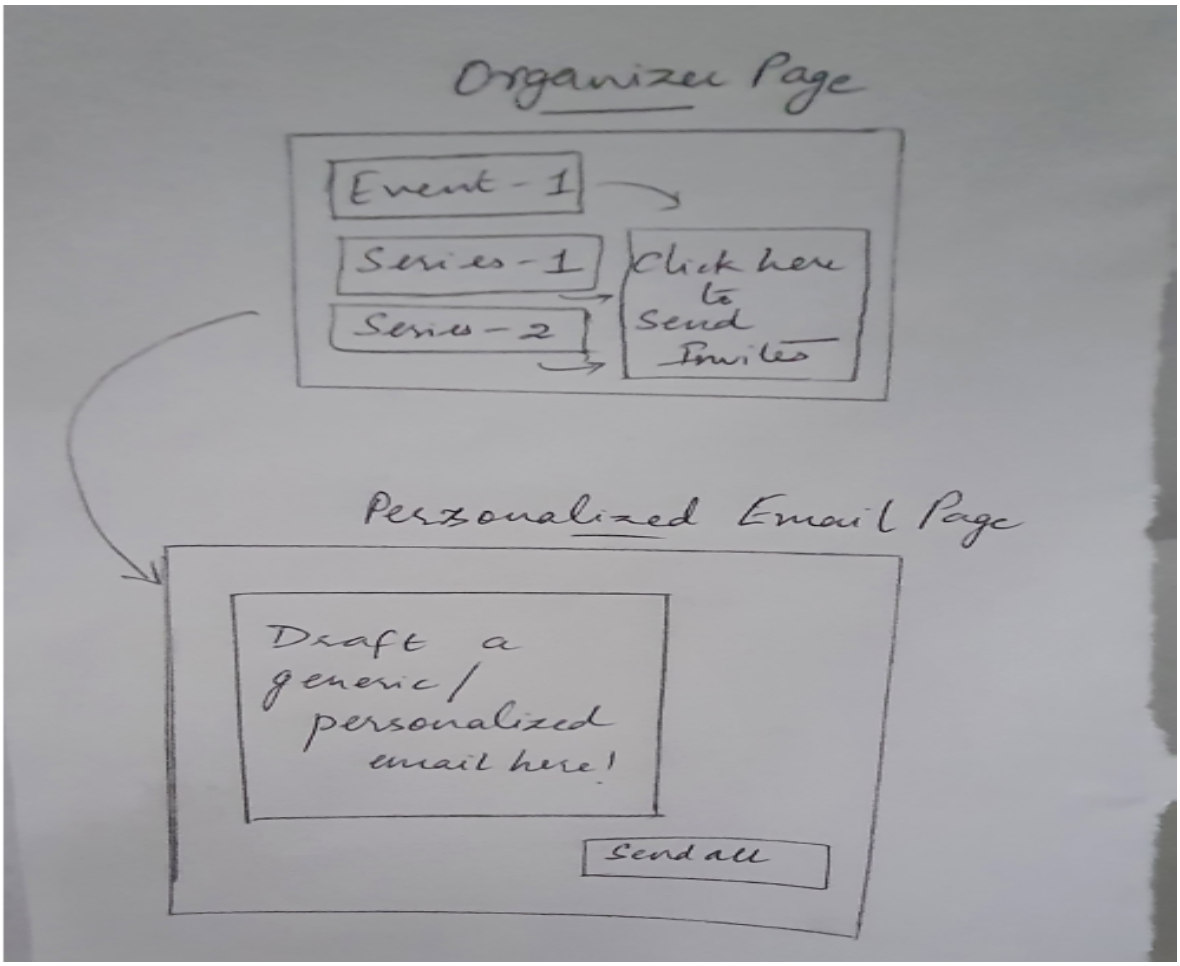


Add People to Event (Repeating Event) Storyboard

This Storyboard describes the way in which an organizer can add people to a repeating event or a series of events.

User Story 5: Sending Email Invites
Status: Implemented
Points: 10

As an event organizer, I want to send personalized invitations to people from a list for my event on skhedule.com. Each invitation should include venue details, and the invitees should have the option to respond with a "yes" or "no," so that I can streamline the invitation process and provide a clear response mechanism. (This functionality is pending approval from client and will be fleshed out further in the next iteration)

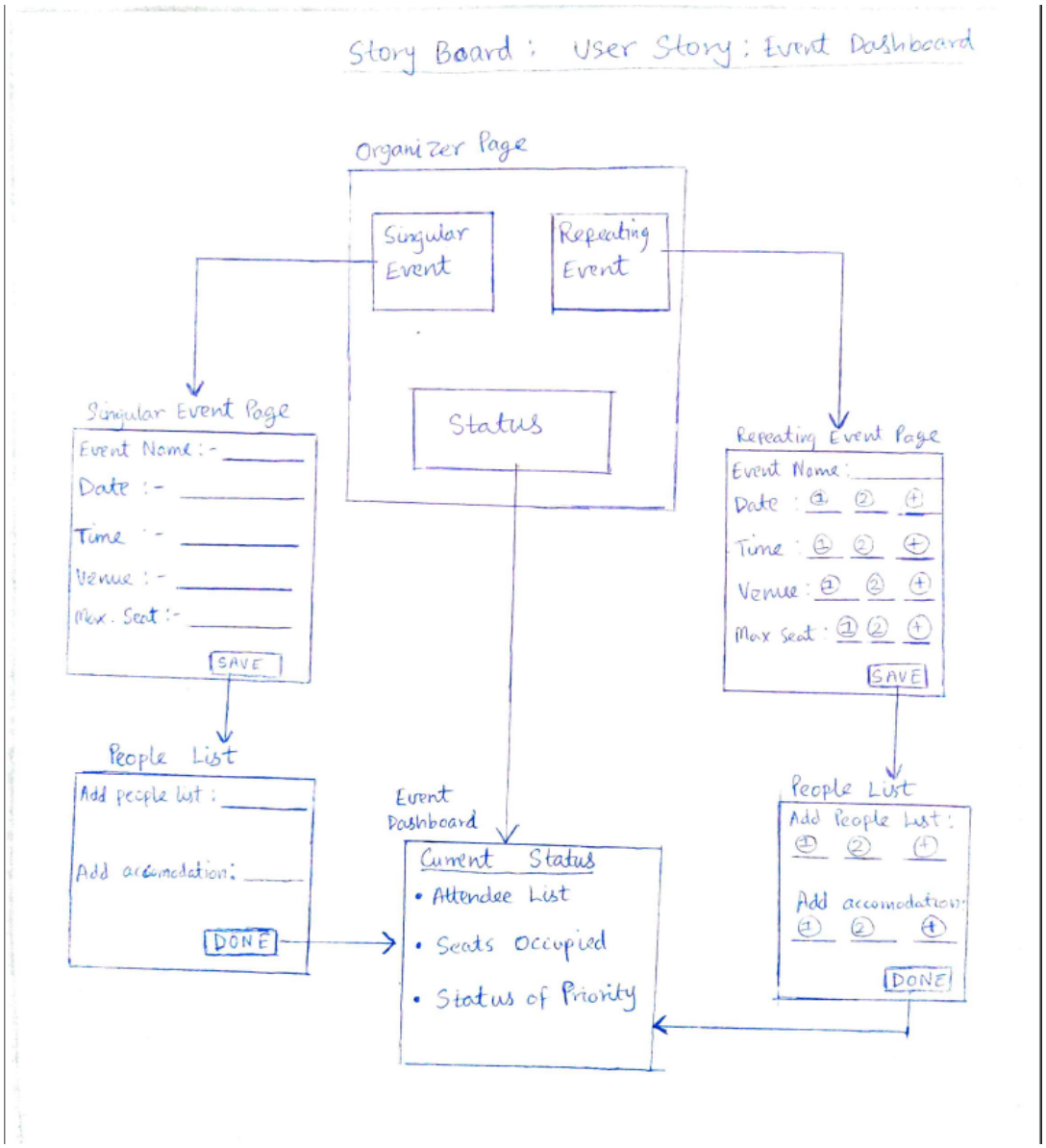


Sending invites storyboard

This Storyboard describes the way in which an organizer can send personalized invite emails to every participant present on their list.

User Story 6: Development of event dashboard
Status: Implemented
Points: 12

As an event organizer, I want access to an event dashboard on skhedule.com. This dashboard should display the current status of my event, including a list of attendees who have responded "yes" or "no," the invitations sent, seats occupied, and the status of any priority lists or waitlists, so that I can keep track of event logistics effectively.



Event Dashboard Storyboard

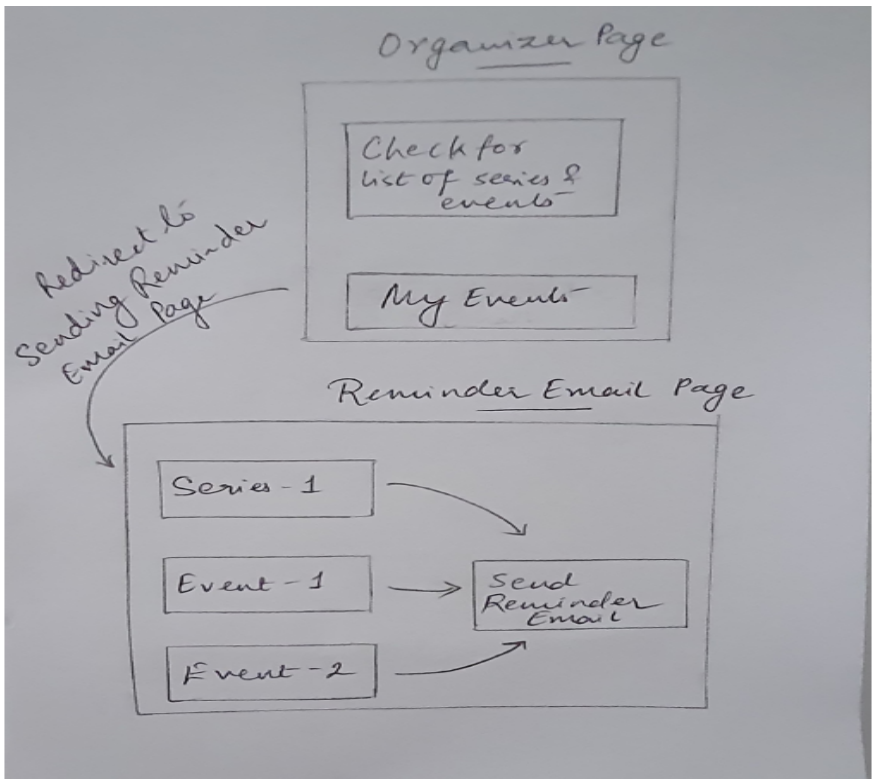
This Storyboard highlights the event dashboard as it should be viewed by the event organizer.

Tasks as part of this User Story:

1. Implement function stubs for testing the email sending functionality
2. Create an event dashboard page that shows the list of events
3. Implement a collapsible event section to display detailed event information when clicked on.

User Story 7: Send reminder emails to attendees who responded yes
Status: Partially implemented
Points: 8

As an event organizer, I want skhedule.com to automatically send reminder emails to users who haven't responded to the event invitation when the event is halfway approaching so that I can ensure that attendees don't forget about the event and encourage timely responses. (This functionality is pending approval from the client and will be fleshed out further in the next iteration)

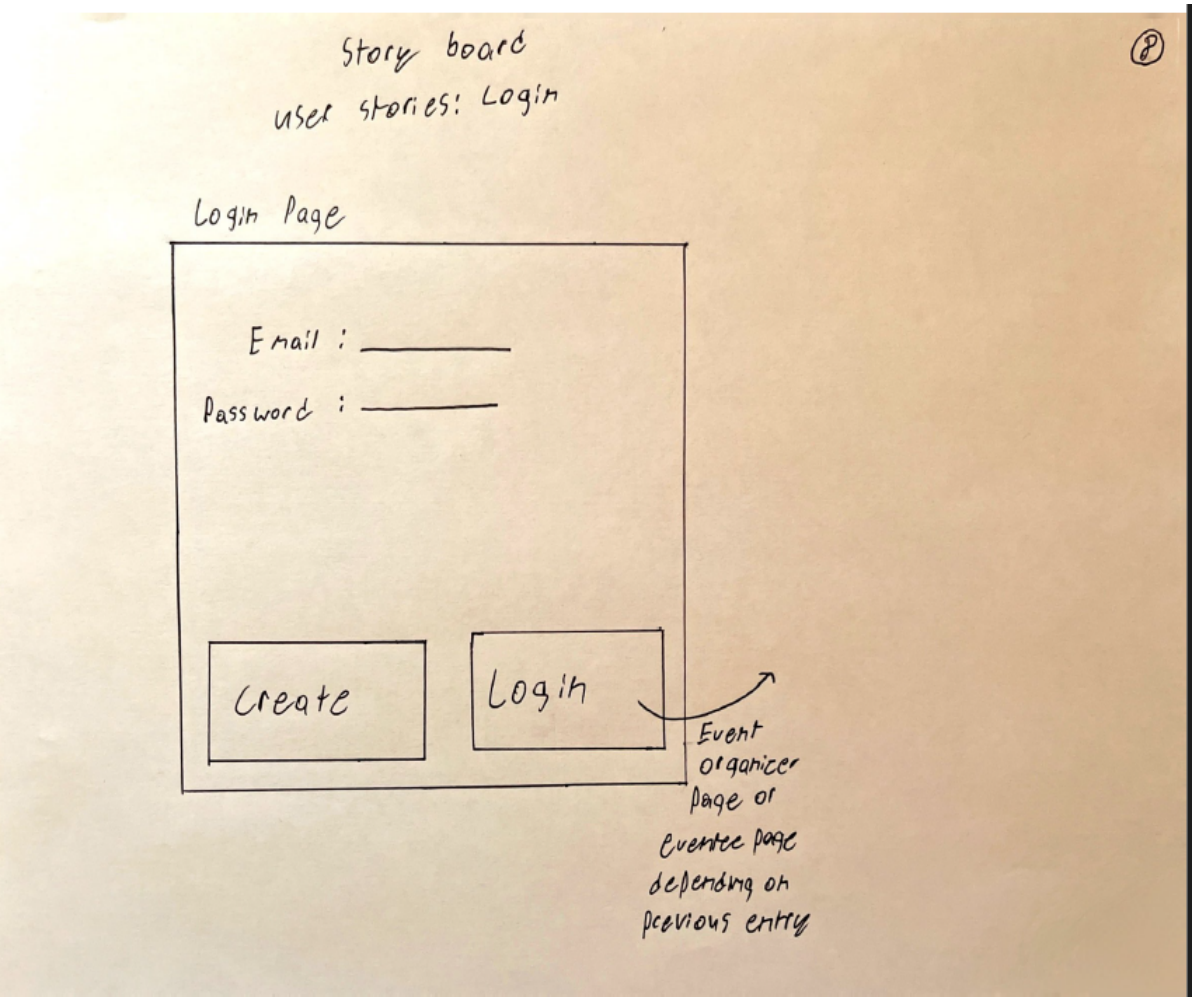


Reminder Storyboard

This Storyboard highlights the reminder that should be sent by the application when an event date is near.

User Story 8: Login Page for Event Organizer
Status: Not implemented
Points: 0

As a user with an account on Skchedule.com, I want to log in to my account via my username and password from the login page, so that I can access the organizer/ eventee page to create/view events.

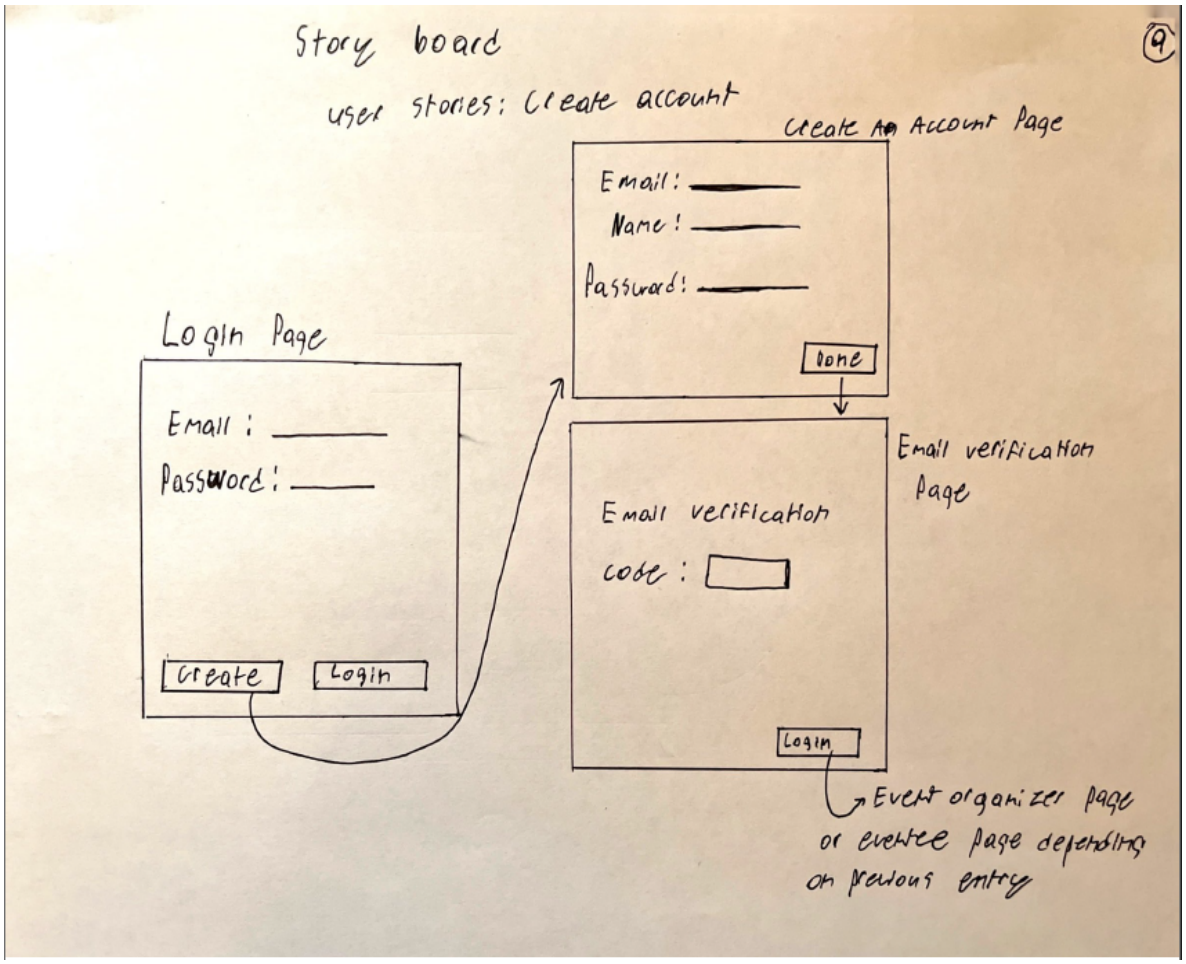


Login Storyboard

This Storyboard highlights the details of the login page. This login page view will be the same for both the event organizer and the event attendee.

User Story 9: Create an Account page for Event Organizer
Status: Not implemented
Points: 0

As a user, I want to create an account on Skchedule.com with my username, password, and email, so that I can then log in to Skchedule.com to create/view events.



Create Account Storyboard

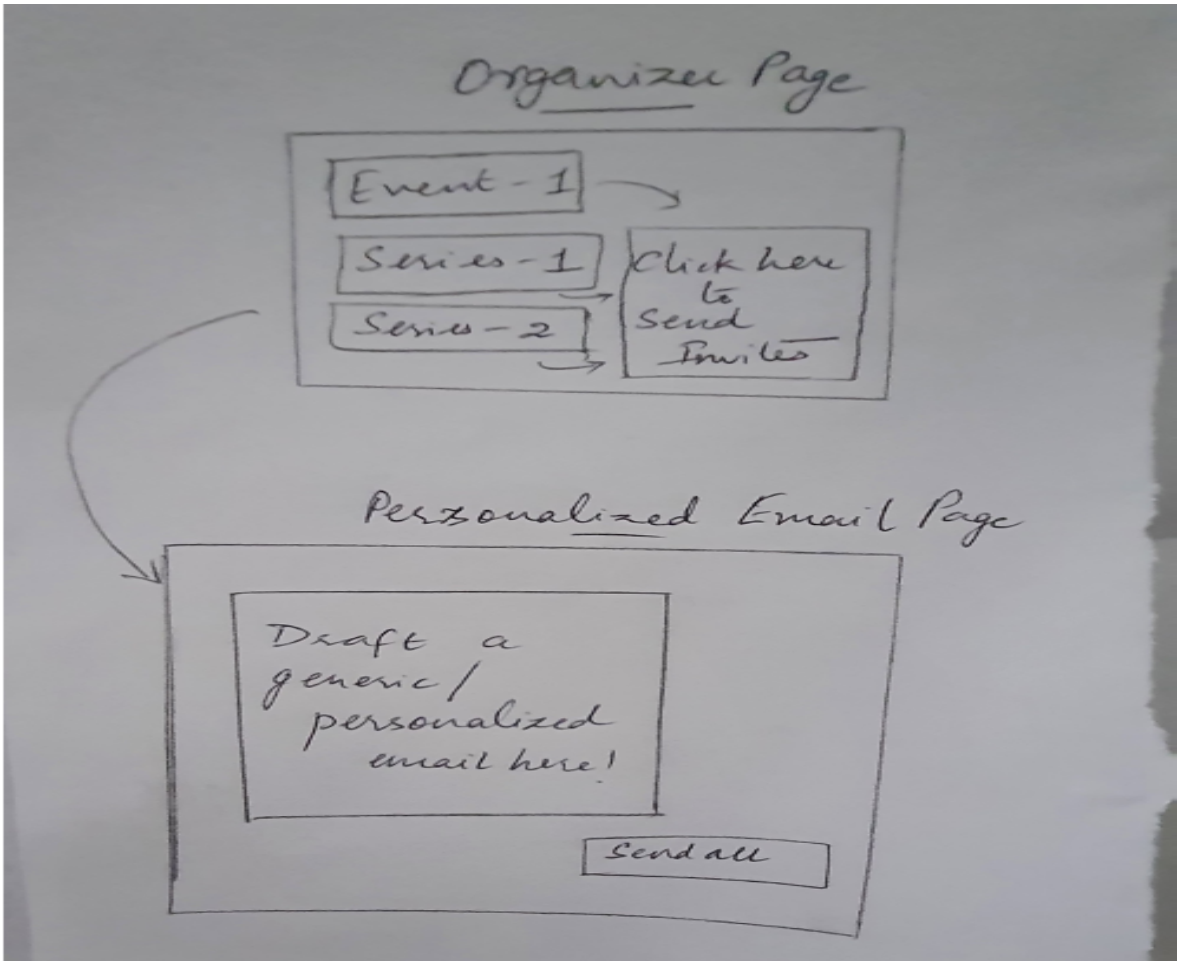
This Storyboard describes the way in which a user can create an account and log in to the web application.

User Story 10: UI Enhancements for the Website
Status: Implemented
Points: 10

As an event organizer and the eventee, I want to view the website with the Bootstrap and CSS styles embedded in it.

User Story 11: Sending and Storing Multiple Invites for Singular and Series Events
Status: Implemented
Points: 12

As an event organizer, I want to send invitations to different people from a list for my event on Skhedule.com. Each invitation should include venue details, and the invitees should have the option to respond with a "yes" or "no," so that I can streamline the invitation process and provide a clear response mechanism. This information is stored in a database so I can view it later.

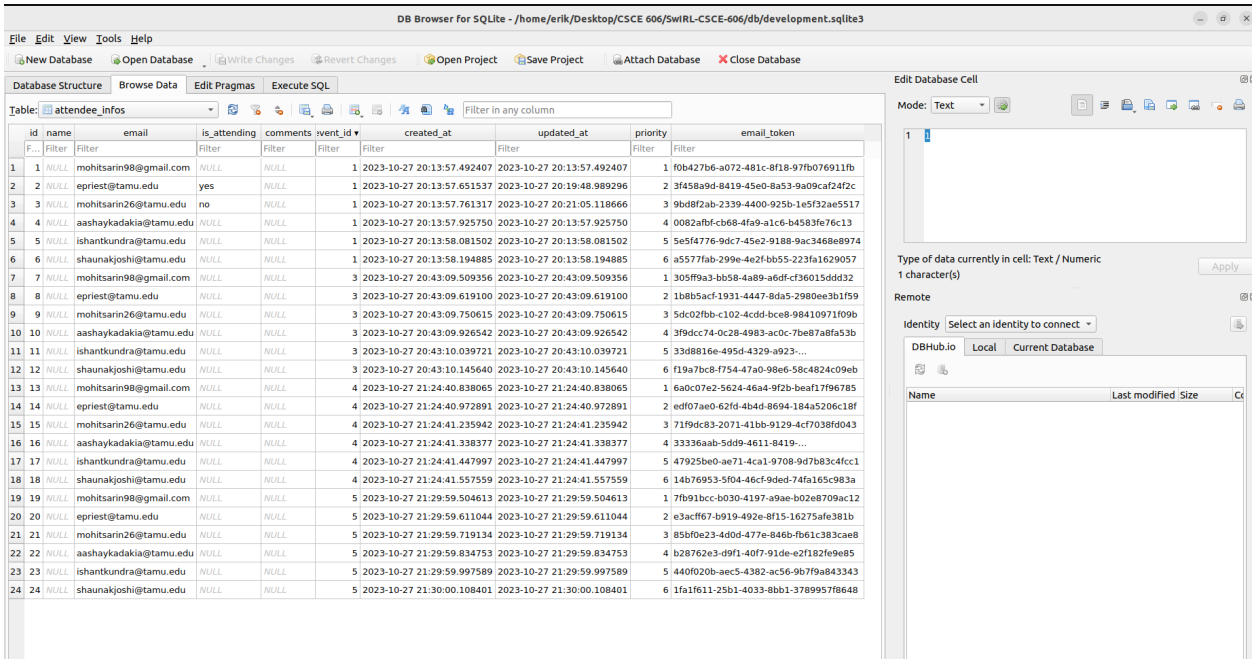


Tasks as part of this User Story:

1. Fix the updation of singular event and add the api to capture (Yes/No) for event invitation
2. Allow multiple users to be added to the event
3. Populate database data to email invitation pages and guarantee the visual effect.
4. Send emails to multiple users

User Story 12: Parsing an uploaded CSV file to extract user emails and priority
Status: Implemented
Points: 14

As an event organizer, I want to upload a csv file to tell Skhedule.com who I want to invite to my event. On the event organizer page, I will have the ability to click a button that says “insert file”, and there I will upload a file with emails and their priority.



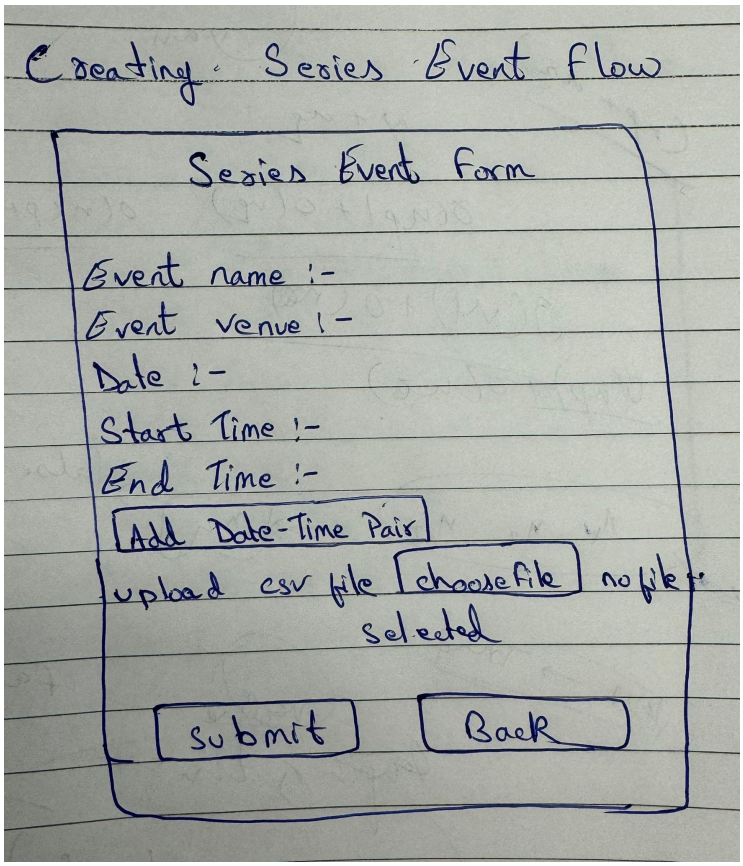
Tasks as part of this User Story:

1. Parse emails of attendees and their priority from the csv file
2. Add a button to upload csv file of attendees

User Story 13: Creating Series Event end to end flow including form page
Status: Implemented
Points: 12

As an event organizer, I want to create a series of events on Skhedule.com with multiple dates, start times, and end times. This feature should allow me to specify the event name and venue for each occurrence in the series, ensuring consistency and ease of scheduling.

Proper data validation is essential to maintain accuracy, and a dedicated database should store these event details for future reference and management.

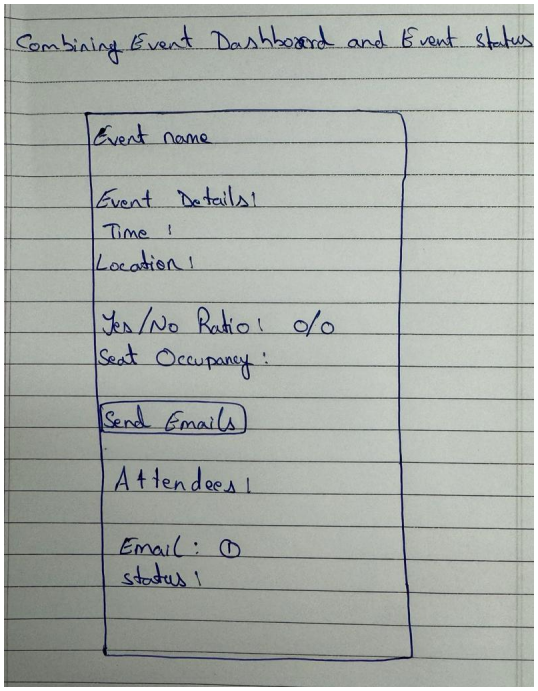


Tasks as part of this User Story:

1. Implement additional columns for date, start time, and end time to accommodate multiple occurrences in the series event.
2. Add fields for the event name and event venue to each series event instance.
3. Ensure data validation for all input fields to maintain accuracy and consistency in event scheduling
4. Develop and integrate a database system for storing and retrieving details of series events.

User Story 14: Combining Event Dashboard and Event Status
Status: Implemented
Points: 5

As an event organizer using Skhedule.com, I want a unified interface where I can view and manage my event's status immediately after creating it. This combined dashboard should display essential details like the event name, date, venue, and the ratio of Yes/No responses from invitees. Additionally, it should include seat occupancy information and a feature to send emails directly from the dashboard. A comprehensive attendance list, showing email IDs along with their respective response status, should also be accessible for efficient event management.



Tasks as part of this User Story:

1. Configure an automatic redirection to the event status page upon the successful creation of an event
2. Ensure the event status page includes all key details: event name, date, venue, seat occupancy,and the Yes/No response ratio of invitees.
3. Implement a 'Send Email' button for easy and direct communication with invitees from the dashboard
4. Display a detailed attendance list on the page, showing each invitee's email ID along with their response status (coming or not).

User Story 15: Implementing Email Sending Logic for Series Event
Status: Implemented
Points: 12

As an event organizer, I want to send emails to the guest who will be able to choose a time slot for their lecture.

Tasks as part of this User Story:

1. Add the Email logic for Series Event to send the emails and capture the API response.
2. Link the database for Series Event to front end form html

User Story 16: Revamping the UI and adding CSS and Bootstrap Changes
Status: Implemented
Points: 10

As a user of the website, I should be able to easily navigate through the web pages and understand their functionality.
Tasks as part of this User Story:

1. Display the current status of the user on the event status page.
2. Fix the Event Status page issues like 1. Add Delete and Edit Button. 2. List all the Data 3. Add Time Slots exists if Applicable (For Series)
3. Fix Frontend issues(made complete new frontend) with CSS and make the styles and background consistent for every HTML page
4. Fix event status page - other various issues

User Story 17: Singular Event Logic for Priority-Based Email Invitations and Reminder Emails for Singular and Series Events
Status: Implemented
Points: 20

As an event organizer on Skhedule.com, I require the ability to send email invitations to invitees based on their priority level for a singular event. This feature should ensure that invitations are sent in an orderly manner, respecting the predefined priority levels. Additionally, robust data validation is crucial to ensure the integrity of priority assignments and the accuracy of the invitation process. Apart from that as an event organizer, I want to send reminder emails for both series and singular events.

	A	B	C
1	Priority	Email	
2	1	mohitsarin98@gmail.com	
3	2	epriest@tamu.edu	
4	3	mohitsarin26@tamu.edu	
5	4	aashaykadakia@tamu.edu	
6	5	ishantkundra@tamu.edu	
7	6	shaunakjoshi@tamu.edu	
8			

Tasks as part of this User Story:

1. Add Reminder Email Functionality for Singular Event
2. Add Initial time expiration for sending the next set of emails in Singular Event. (Add this field as input to form as well)
3. Fix Singular Event Priority Sending Logic to include only "No" condition

User Story 18: Prepare Final Report and Presentation/Demo
Status: Implemented
Points: 12

Prepared a final project report and recorded demo and presentation.

Unfinished/ Unimplemented User Stories:

Our team's user stories went through many changes through the iterations. They started off broad, but as the iterations progressed we had to narrow them down and break them up into smaller stories. User stories 1-9 were our initial thoughts of the project, however user stories 8 and 9 were not implemented at all.

User Story 8 and 9 (**Login and Account Creation**):

Originally, the team planned on adding a login page for event organizers so they could view their own events whenever they wanted, but time constraints prevented implementation.

User Story 7 (Reminder Email):

User story 7 was partially implemented; while our team originally planned to have reminders sent out to invitees automatically before the event, we ultimately implemented an option for the event organizer to manually send out reminders if they desired. User stories 10-17 were created and implemented as Skhedule.com became more fleshed out.

Additionally, a good to have user feature would be to add a time for invalidating the invite sent out to the users. This would allow the users with lower priority to receive the invites to the events earlier.

Roles

During the course of the project some roles were swapped due to desire amongst team members to work in another role. In addition, we tried to set a new scrum master each week to allow everyone a chance to be a scrum master. The only constant was the selection for Product Owner as we did not want to confuse our communication with the client with various emails. The following table shows who did what for each iteration:

Iteration 0:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Scrum Master / Full-Stack Dev	Front-end Dev	Back-end Dev	Back-end Dev	Front-end Dev

Iteration 1:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Full-Stack Dev	Front-end Dev	Back-end Dev	Full-Stack Dev	Scrum Master / Full-Stack Dev

Iteration 2:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Scrum Master / Full-Stack Dev	Front-end Dev	Full-Stack Dev	Full-Stack Dev	Full-Stack Dev

Iteration 3:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Full-Stack Dev	Front-end Dev	Full-Stack Dev	Scrum Master / Full-Stack Dev	Full-Stack Dev

Iteration 4:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Full-Stack Dev	Scrum Master / Front-end Dev	Full-Stack Dev / Testing	Full-Stack Dev	Full-Stack Dev

Iteration 5:

Mohit Sarin	**Shaunak Joshi**	**Ishant Kundra**	**Chong Wen**	**Aashay Kadakia**	**Erik Priest**
Product Owner / Full-Stack Dev	Scrum Master / Full-Stack Dev	Front-end Dev	Full-Stack Dev / Testing	Full-Stack Dev	Full-Stack Dev

Scrum Iteration Summary

Iteration-0:

Points Completed in this Iteration: 30

1. Initiated project foundation:
    - Finalized basic architecture, team roles, and initial user stories based on client requirements.
    - Prioritized and created Low-Fidelity (Lofi) Mockups for user stories.
2. Project setup and deployment:
    - Built and deployed the project on Heroku.
    - Established a GitHub repository for version tracking.
3. Story visualization:
    - Created storyboards for all user stories.
    - Developed a high-level project summary, encompassing customer needs and application alignment.

Iteration-1:

Points Completed in this Iteration: 40

1. Front-end and back-end development:
    - Developed the main page in simple HTML format.
    - Created a form for the Singular Event Page based on client discussions.
    - Developed the initial Series Event Page.
2. Database setup:
    - Created different databases for Production, Test, and Dev environments.
    - Added required tables to store form, email, and invitee attributes information.
3. Feature testing:
    - Added RSpec test cases for the developed features.

Iteration-2:

Points Completed in this Iteration: 43

1. UI enhancement:
    - Improved website UI using Bootstrap and CSS.
2. Back-end Changes:
    - Implemented changes suggested by the client on the Series Event Page.
    - Included SMTP server for automated emails in Development and Production environments.
3. Database refinement:
    - Improved the database schema, creating two tables to store event and attendee information separately.
4. Feature testing:
    - Added RSpec and Cucumber test cases for updated and newly added features.

Iteration-3:

Points Completed in this Iteration: 58

1. Enhanced functionalities of Singular Events:
    - Allowed event organizers to upload a CSV file for creating an invitee list.
    - Captured "Yes" and "No" responses from invitees.
    - Enabled event organizers to send emails to all invitees using the "Send Email" button.
2. UI improvements:
    - Improved UI button views and colors based on client feedback.
    - Implemented a collapsible event section for detailed event information.
3. Feature testing:
    - Developed an event dashboard page to view information on all upcoming events.
    - Added new RSpec and Cucumber test cases for the features developed in this sprint.

Iteration-4:

Points Completed in this Iteration: 39

1. Form creation and validation:
    - Created Series Event form and updated form information in the database.
    - Implemented data validation in the front and back end for all fields in Singular and Series Events.

2. Logic and user experience:
    - Updated the Event Dashboard page to include new fields discussed in client meetings.
    - Enhanced logic for Singular Event with Priority-Based Email Invitations.
3. Feature testing:
    - Improved user experience and UI for easy user interaction.
    - Added new RSpec and Cucumber test cases.

Iteration-5:

Points Completed in this Iteration: 36

1. Completion of end-to-end flow:
    - Added Event Sending feature for Series events.
    - Improved priority-based Email Sending Invitations for Singular Events, completing the end-to-end flow.
2. Additional functionalities:
    - Added Reminder Email Functionality for Event Organizers in both Singular and Series events.
3. Feature testing:
    - Added new RSpec and Cucumber test cases.

Story points completed memberwise:

Team Member Efforts

Overall Story Points Distribution:

Name	**User Story Points**	**Effort** (points/average of points)
Erik Priest	41	100%
Aashay Kadakia	40	98%
Mohit Sarin	42	102%
Shaunak Joshi	41	100%
Chong Wen	41	100%
Ishant Kundra	41	100%

Team Member Efforts (Iteration 0)

Name	**User Story Points**	**Effort** (points/average of points)
Erik Priest	4	80%
Aashay Kadakia	6	120%
Mohit Sarin	4	80%
Shaunak Joshi	4	80%
Chong Wen	6	120%
Ishant Kundra	6	120%

Team Member Efforts (Iteration-1)

Name	User Story Points	Effort
Erik Priest	7	105.11%
Aashay Kadakia	6	90%
Mohit Sarin	8	120.12%
Shaunak Joshi	8	120.12%
Chong Wen	6	90%
Ishant Kundra	5	75.07%

Team Member Efforts (Iteration-2)

Name	User Story Points	Effort
Erik Priest	8	104.44
Aashay Kadakia	3	35.71
Mohit Sarin	9	119.04
Shaunak Joshi	8	104.76
Chong Wen	8	107.14
Ishant Kundra	7	100

Team Member Efforts (Iteration-3)

Name	**User Story Points**	**Effort** (points/average of points)
Erik Priest	9	93.75
Aashay Kadakia	14	144.93
Mohit Sarin	8	82.81
Shaunak Joshi	8	82.81
Chong Wen	9	93.75
Ishant Kundra	10	103.57

Team Member Efforts (Iteration 4)

Name	**User Story Points**	**Effort** (points/average of points)
Erik Priest	7	107.6
Aashay Kadakia	5	77.0
Mohit Sarin	7	107.6
Shaunak Joshi	7	107.6
Chong Wen	6	92.3
Ishant Kundra	7	107.6

Team Member Efforts (Iteration 5)

Name	**User Story Points**	**Effort** (points/average of points)
Erik Priest	6	100
Aashay Kadakia	6	100
Mohit Sarin	6	100
Shaunak Joshi	6	100
Chong Wen	6	100
Ishant Kundra	6	100

Customer Meetings

Iteration 0:

We contacted our client (Jonathan Tan) on 9/11/2023 and sent over these questions but couldn't meet our client on our usual meeting day Tuesday (9/12/2023) as they didn;t have the answers to our questions at the moment and were going to regroup with their team for this. We tried to reschedule this meeting to some other day as per their convenience but we haven't heard back from them since. As a result, some of our questions are still unanswered. These are the questions we sent over to Jonathan on 9/11/2023.

List of Questions (9/11/2023)

1. What front-end framework should we use for Skhedule.com? Is there any preference or should we go forward with any framework that we are familiar with?
2. Who are the stakeholders? Would that be SwiRL or the website would be outsourced to a third party as well?
3. Is this meant to be a public website? So for public events, anyone who happens to see it can sign up? Also, for private events, only the people who are invited can see the invite.
4. What should we do if it is an event without an invite list? In this case, we should make the event public and let people register till we reach the maximum capacity.
5. Are attendees allowed to mention notes like if the attendee is allergic to some food? Or an Event Coordinator should put that up as a note. In general, should we allow attendees to add notes on restrictions?
6. If someone brings one more person with them to an event? In that case, what should be the priority list for the added person? Also, in that case, how does the waitlisting algorithm work, if the event is left with only one seat, shall the user be allowed to add one more person, or they should not be allowed to do so?
7. Do we need the users to have an account, where they can view the events they are invited to? If yes, do we need to make them sign up for this account and then store their login details in a database preferably?

We later had a meeting with Jonathan, where we got some of these questions answered:

Front-end Framework: Choose a front-end framework based on your team's familiarity and alignment with the back-end technology stack.
Stakeholders: Identify SwiRL as the primary stakeholder.
Public vs. Private Events: Make it a public website.
Events Without Invite Lists: Yes, until the seats are filled.
Attendee Notes: Allow attendees to add notes.
Additional guests: The additional guests will enter through the same gateway; no priority.
User Accounts: Users simply accept the invite by entering the needed details

Iteration 1:

1. Place: Zoom Meeting
2. Date and Time: 09/21/2023 9 am with Jonathan, (09/28/2023) with Jonathan and Marcus

Minutes of the meeting (09/21/2023):

- Explained the Lo-Fi Mockup UI to the client. The client approved the Lo-Fi Mockup.
- No priority list if the invitee wants to bring a '+1' member (Outside project scope as of now)
- On the website, we should provide a basic filter over the list of public events.
- The client requested a digital copy of the Lo-Fi Mockup and suggested making it using wireframe websites, to make it easy to discuss with all the stakeholders.
- Additional notes should be dietary restrictions, pronouns, and any other info that the invitees need to provide.
- Invitee should receive the event link without signing in, client agrees with this.
- The goal of the dashboard is for the organizer to have a good view of the attendees.
- How many reminder emails should be sent - Depends on the organizer but there should be 1 standard reminder.
- Is there a need for a different UI for the organizer and the invitee on Skhedule.com? - One role for email, Just the event organizer needs to sign up …..This is something Jonathan plans to confirm with their team and will get back to us in the next meeting.
- Adding the client to our Slack and Google Drive so that they can access all the project-related documents
- Focus on getting core features done for this iteration, e.g. setting up an event and sending out invites

Minutes of the meeting (09/28/2023):

- Showed the client the updated digital mockup that they requested.
- Gave them a basic demo of the web pages that we have designed so far.
- There is another version of this project in Python/Django which the client already has developed.
- Discussed with Jonathan the database issues that we are currently having.
- Requested feedback on the Lo-Fi Mockup and we had an in-depth discussion with Marcus about the project expectations. (A few changes to the initial requirements have been recommended by Marcus)
- At dinner parties, we will send out invites to all the attendees(max number of slots), whereas in the case of speaker events, we only need to send one invite at once to the speaker.
- For the speaker events, who are the invitees? ……The speakers are the invitees in this case.

- Do we also need to send emails after the speaker has accepted the invitation? Jonathan suggested we use the dinner party scenario post the speaker accepted the invite.
- Date and time needed on speaker events….No venue is needed.
- Once the speaker is selected, how will we get the people list who are going to attend this? Will the professor provide this to us or should we use a random list? We can create a CSV with 5 people to test our functionality in our testing scenario.
- Any free service for SMTP? SendGrid and MailGun…..Marcus insisted on using SendGrid with Rails.
- Flesh out the roles in the groups in a more detailed way like the project manager, database architect, UI/UX dev, and backend dev.
- Marcus will send us the older Python/Django repo link for this project. (Still pending)

Iteration 2:

Place: Zoom Meeting
Date and Time: 10/19/2023, 9:00 am with Jonathan

Minutes of the meeting (10/19/2023):

- Provided the working Demo of the project till Iteration 2.
- Shared the iteration 3 goals and prioritized the stories for this iteration.
- Marcus told us to focus on improving our database, uploading the CSV with the priority and the list of invitees, and sending emails.
- They also suggested including max capacity of seats on the singular event form.
- Jonathan suggested including and capturing Yes and No responses from the invitees through email.
- Marcus suggested a few corrections for iteration report 2 and asked us to remove irrelevant things such as the QR scan, and login page as these requirements are no longer required and prioritized.

Iteration 3:

Place: Zoom Meeting
Date and Time: 10/19/2023, 9:00 am with Jonathan
Minutes of the meeting (10/19/2023):

- Provided the working Demo of the project till Iteration 2.

- Shared the iteration 3 goals and prioritized the stories for this iteration.
- Marcus told us to focus on improving our database, uploading the CSV with the priority and the list of invitees, and sending emails.
- They also suggested including max capacity of seats on the singular event form.
- Jonathan suggested including and capturing Yes and No responses from the invitees through email.
- Marcus suggested a few corrections for iteration report 2 and asked us to remove irrelevant things such as the QR scan, and login page as these requirements are no longer required and prioritized.

Iteration 4:

Date and Time: 11/02/2023, 9:00 am with Jonathan
Minutes of the meeting (11/02/2023):

- Demonstrated the progress of the project up to Iteration 3 to Jonathan.
- Discussed Jonathan's concerns about the functionality of the 'Destroy' and 'Edit Event' buttons.
- Agreed that the default attendee response should be set to 'pending response' on the event status page
- Identified the implementation of series/repeating events as a high-priority task.
- Reviewed the Iteration 3 document with Jonathan for further insights
- Emphasized the need for refining CRUD (Create, Read, Update, Delete) operations.
- Clarified doubts regarding the prioritization in the CSV file, deciding to align with the standards of other accounting software (considering 0 as the highest priority).
- Addressed uncertainties about the email reminder system.
- Jonathan emphasized focusing on the minimum viable product, specifying two main priorities:
    - 1. Ensuring the organizer has an account for sending out emails and capturing responses.
    - 2. Prioritizing the speaker event feature and perfecting its algorithm.

Iteration 5:

Place: Zoom Meeting

Date and Time: 16/11/2023, 9:00 am with Jonathan
Minutes of the Meeting (16/11/23):

- Discussed all the latest developments and provided the working demo of the project to Jonathan and Marcus.
- Jon suggested keeping a Heroku scheduler and keeping it under an hour or something, if we plan to implement horizon time for email expiry. Create a shell script that runs it using rake on Ruby code. Marcus said that we need to run it every 10 minutes. If we weren't doing Heroku, we could have used Cron. Still, the requirement was not clear and hence, we didn't implement this for this iteration.
- Creation of accounts is something Jonathan brought up and we told them that we might not be able to get it done in the next 2 weeks.
- Jonathan inquired about the event dashboard and we told him about the latest changes that we made to this page in iteration 4 and provided them with a demo of the new UI of the website.
- The default value of email in the Event Dashboard should be Pending and not attending.
- Make UI elements look more consistent using Bootstrap and CSS.
- UI changes were shared. Jonathan said that he wouldn't go for a black color here but Marcus seemed to agree. They said that the color scheme is something up to us and we can experiment a bit with it.
- If we want to use a background image, we can use media queries and then use it in the viewport. Marcus said that instead of using images, we should stick to a plain background.
- Set the priorities straight for iteration 5: full singular event and speaker event workflow, consistent UI, good test coverage, and reminder email.

BDD / TDD Process

Throughout our development process, we have adhered to an iterative testing approach. Across all six iterations, we introduced new features and meticulously crafted test cases associated with each enhancement. Below is a detailed account of our testing endeavors during these iterations, highlighting the integration of Behavior-Driven Development (BDD) with Cucumber and Test-Driven Development (TDD) with RSpec.

In every iteration, we are gathering the requirements from the client and maintaining a priority for the user stories that need to be completed in the given sprint. For picking up the priority user stories, we kept in mind the initial user requirements and which features are most important from the user perspective, and hence, we will deliver them in initial iterations. Also, we developed the test cases hand-in-hand with the development.

1. Understanding BDD and TDD: BDD, which concentrates on the system's behavior from the user's perspective, entails describing expected behaviors using natural language specifications. In contrast, TDD necessitates writing tests before code implementation, following the "Red-Green-Refactor" cycle. This entails writing a failing test, implementing code to pass the test, and subsequently refactoring the code while ensuring the tests still pass.

2. BDD Process with Cucumber:
- Write Feature Files: Utilizing Cucumber, we drafted feature files to articulate high-level behaviors in a human-readable format.
- Step Definitions: Implementation of step definitions in Ruby aligned with the steps outlined in the feature files.
- Run Cucumber: Execution of Cucumber tests to validate that the application's behavior aligns with the specified features.
- Iteration Testing: Application of these Cucumber tests across all six iterations, each concentrating on distinct features introduced during that specific phase.

3. TDD Process with RSpec:
- Our TDD process mirrored that of Cucumber.
- Write a Failing Spec: Definition of behavior through specifications using RSpec, constituting the "Red" phase.
- Write Code to Pass Spec: Implementation of the minimal code required to make the spec pass, signifying the "Green" phase.
- Refactor Code: Subsequent to the spec passing, the code underwent refactoring, ensuring all specs continued to pass.
- Iteration Testing: Execution of RSpec tests throughout all six iterations, encompassing various unit tests for features introduced during each iteration.

4. Integration of RSpec and Cucumber:
- Combination of RSpec for unit testing and Cucumber for higher-level acceptance testing.

- Ensured comprehensive coverage with both RSpec and Cucumber; RSpec for more granular, low-level tests and Cucumber for high-level, end-to-end tests.
- More RSpec test examples were utilized compared to Cucumber, owing to the diverse routes available for testing as opposed to the features specified for Cucumber test cases.

5. Updation and Maintenance of Test Cases:
   - Regular and systematic review and update of tests with the evolution of the codebase. This practice ensured tests accurately reflected the intended behavior and facilitated the identification of regressions.

Benefits and Challenges of BDD and TDD:

Benefits of BDD:
1. Clear Understanding of Requirements.
2. Improved Test Coverage.
3. Facilitates ease in updating tests when the codebase changes.

Challenges of BDD:

1. Writing different test scenarios for the same feature can be challenging.
2. There may be an overemphasis on syntax.

Benefits of TDD:

1. Easy and Early Bug Detection.
2. Design Improvement.
3. Improved Test Coverage.

Challenges of TDD:
1. Maintaining and updating a large number of test cases can be challenging at times.

In conclusion, following the BDD and TDD process, helps us to better manage the development, test cases and the user needs together. We were substantially benefited in terms of collaboration, early bug detection, and improved code design.

Configuration Management Approach

In our version control system (VCS) setup, Git and GitHub serve as the cornerstone of our collaborative development and code management process. Git, with its distributed version control capabilities, enables us to track changes made across various aspects of our project, including codebase modifications, documentation updates, and design alterations throughout different iterations. GitHub, being our chosen hosting platform for Git repositories, not only facilitates version control but also enhances collaboration among individuals involved in the project

In our code management strategy, we maintain three primary branches: development, testing, and main. The development branch serves as a controlled environment for testing changes before integration into the main branch, acting as a crucial failsafe step before deploying code to production. Within the testing branch, comprehensive test cases are meticulously outlined, and the branch is subsequently merged with the development branch to incorporate up-to-date Rspec and Cucumber test cases aligned with the latest code changes in the iteration. Once changes are thoroughly validated in the development branch, we proceed with deployment to the main branch.

Automation plays a pivotal role in our workflow, particularly on the main branch, where we have established an automated deployment process to Heroku. Consequently, each commit to the main branch triggers an automated deployment to Heroku, ensuring a seamless integration of changes into our production environment.

Our project repository on GitHub is complemented by the use of GitHub Projects to meticulously track the progress of tasks. At the onset of each iteration, tasks are created and added to the Backlog panel. Upon finalization by our customer, these tasks transition to the In Progress view, signaling the commencement of work. GitHub Projects facilitates the assignment of story points and assignees to each task, providing a structured means of monitoring the implementation of individual story points throughout the iteration. At the conclusion of each iteration, completed tasks are transitioned to a dedicated view for that specific iteration, allowing for a comprehensive review of all undertaken and completed tasks. Any remaining tasks in Backlog, In Progress, or In Review are seamlessly transitioned to the next iteration.

Moreover, our deployment history on Heroku reflects a total of 62 deployments from our main branch. To streamline and categorize our progress, we have created five major tags corresponding to each iteration, labeled iteration1 through iteration5. These tags encapsulate the latest code changes, comprehensive documentation containing customer meeting summaries, tasks undertaken during the iteration, Behavior-Driven Development (BDD) and Test-Driven Development (TDD) artifacts for the current iteration, and design changes slated for implementation in the ongoing iteration. This meticulous tagging system ensures a structured and traceable record of our project's evolution over multiple iterations.

Heroku Deployment Steps and Issues Faced

Overall, our Heroku deployment was smooth. We initially deployed our static app in Iteration 0, and thereafter, we pushed changes from GitHub directly to Heroku via the CI/CD pipeline linking GitHub and Heroku. One small issue was that we were not initially using the correct database on the Heroku server. However, we later recognized the mistake and updated the database on Heroku to one that best suits our needs—PostgreSQL. All of these adjustments were made in Iteration 0. Subsequently, there were no major issues with deployment on Heroku. The Heroku deployment steps have been outlined in detail in the 'Read Me' section of our Git repository.

There are two known issues that you might face while running the project. Herr are the troubleshooting steps for those issues:

One known issue while running the application locally is that when an invitee tries to accept or reject an event invitation, the API will redirect to prod, and hence, the page will not be reachable (as you are running locally). To overcome this, before capturing your response through email, change the link that the email redirects to (right-click and copy the link address) before clicking "Yes" or "No".

To overcome this, make sure to remove the defense URL, if you are on a protected network.
For Instance, these are sample URLs for capturing responses:
ProdURL:
http://skhedule-9d55cf93012e.herokuapp.com/events/1/yes_response?token=73eebd46-76a6-4edc-a3ae-0a2e8670ca17
LocalURL:
http://127.0.0.1:3000/events/1/yes_response?token=73eebd46-76a6-4edc-a3ae-0a2e8670ca17

Second, make sure you ask for the master.key for running the Gmail SMTP server locally. Since it is sensitive information, we have not posted the key on the git repository. Please contact the Product Owner (mohitsarin26@tamu.edu) for the details of the existing master.key if you are running locally.

If you are deploying the app in Heroku, you need to re-create a rails credentials file with the variables names USER_NAME and PASSWORD which is the corresponding password and username for the Gmail SMTP.
Here are the steps to create a key to run SMTP using the below commands.
heroku config:set RAILS_MASTER_KEY=`cat config/master.key`
heroku rails db:migrate

The More information can be found in the Read Me section of our GitHub Repository.

Other tools and Gems:

1. **rspec:** RSpec, a testing framework for Ruby, facilitates behavior-driven development (BDD) by enabling developers to articulate specifications and test cases. It not only elevates code quality but also establishes a structured approach to testing.
2. **rspec-rails**: Specifically tailored for Rails applications, RSpec-Rails extends the capabilities of RSpec to seamlessly integrate with the Rails framework. It introduces Rails-specific matchers and integrations, streamlining the testing processes inherent in Rails projects.
3. **Simplecov**: SimpleCov stands as a code coverage analysis tool for Ruby, aiding developers in comprehensively tracking the code covered by tests. It generates detailed coverage reports, offering insights into areas of code that may necessitate additional testing.
4. **'csv'**: The 'csv' module, an integral part of Ruby's standard library, simplifies the reading and writing of Comma-Separated Values (CSV) files. It proves to be a convenient tool for manipulating CSV data, particularly useful in handling tabular data.
5. **'securerandom'**: The 'securerandom' module, an inherent component of Ruby's standard library, provides a secure means to generate random numbers or strings. Frequently employed for tasks like generating unique identifiers or cryptographic tokens, it enhances security in various applications.
6. **module 'calendarhelper':** The CalendarHelper module in Ruby on Rails serves the purpose of generating Gmail invites for events. This module encapsulates the logic for constructing Gmail invite links based on event details, fostering the development of clean and maintainable code.

Github repository details

Link to our Github repository: https://github.com/SwIRL-CSCE-606/SwIRL-CSCE-606

1. Within our primary branches, specifically the main and development branches, a pivotal component of our project architecture resides in the "**app**" folder. This directory serves as the core repository for our application's critical codebase, housing files integral to the functionality and structure of the software. Notably, within the "app" directory, one can find files about controllers, views, and models – the foundational elements that collectively define the application's behavior and user interface.

   Situated within the **controllers** subfolder, our primary events_controller functions as the focal point for core logic about event management. It encapsulates essential algorithms and functionalities related to singular event creation, series event creation, event deletion, and event updating. This controller orchestrates the intricacies of inviting attendees based on a priority system, employing sophisticated mechanisms to capture

user responses through API interactions for seamless processing of affirmative or negative replies.

Within the **models** subdirectory, we house a comprehensive set of database tables integral to our project. Noteworthy entities include the event, event_info, attendee_info, time_slots, and application_record tables. Each of these tables plays a distinct role in structuring and organizing the underlying data architecture of our application, contributing to the seamless functioning and persistence of critical information.

Under **views**, we have all the web pages that are being used in our application. These are HTML web pages with CSS, Javascript, and Bootstrap used for dynamic content.

Under **mailers**, we have application_mailer and event_remainder_mailer, which contain code related to SMTP setup and sending emails to users for series and singular events

2. In the **config** folder, we have information related to all the environment variables, env files for SMTP credentials that are required for sending emails, routes for all the functions in the application, and the YAML files for database, cucumber, etc.

3. In the **db** folder, we have info related to the database, it's the schema, the seed data that we were initially using and all the database migrations that we performed later on throughout the development.

4. In the **documentation** folder, we have the documentation tar files for all the iterations from i0 to i5.

5. In the **features** folder, we have all the step definitions and cucumber tests for our application.
6. In the **lib** folder, we have added rake files for cucumber tests and email reminders.

7. In the **spec** folder, we have all the Rspec tests that we have written for our application. These are the tests for the controllers, models, and the mailer that we have in our app folder. We have done extensive testing of our application using Rspec and we have all our 47 tests passing in the main as of now.

8. All the steps required to get our application running and to deploy it are mentioned in great detail here: https://github.com/SwIRL-CSCE-606/SwIRL-CSCE-606/blob/main/README.md. We have also added steps to troubleshoot and contact us in case of any issues that arise in the project.

Link to our resources

Deployed App - https://skhedule-9d55cf93012e.herokuapp.com/

GitHub Repo - https://github.com/SwIRL-CSCE-606/SwIRL-CSCE-606

Project Management - https://github.com/orgs/SwIRL-CSCE-606/projects/3/views/1

Slack channel - https://join.slack.com/t/swirlcsce606/shared_invite/zt-22rtanhvr-5DrTEWAFPi5E_YjEInIBhw

Presentation and Demo - https://www.youtube.com/watch?v=B_rrLmjhplM