**Web App Project - Planiac**

**Feature Details:**

Your social event planning web app should include, and will be assessed on, the following features:

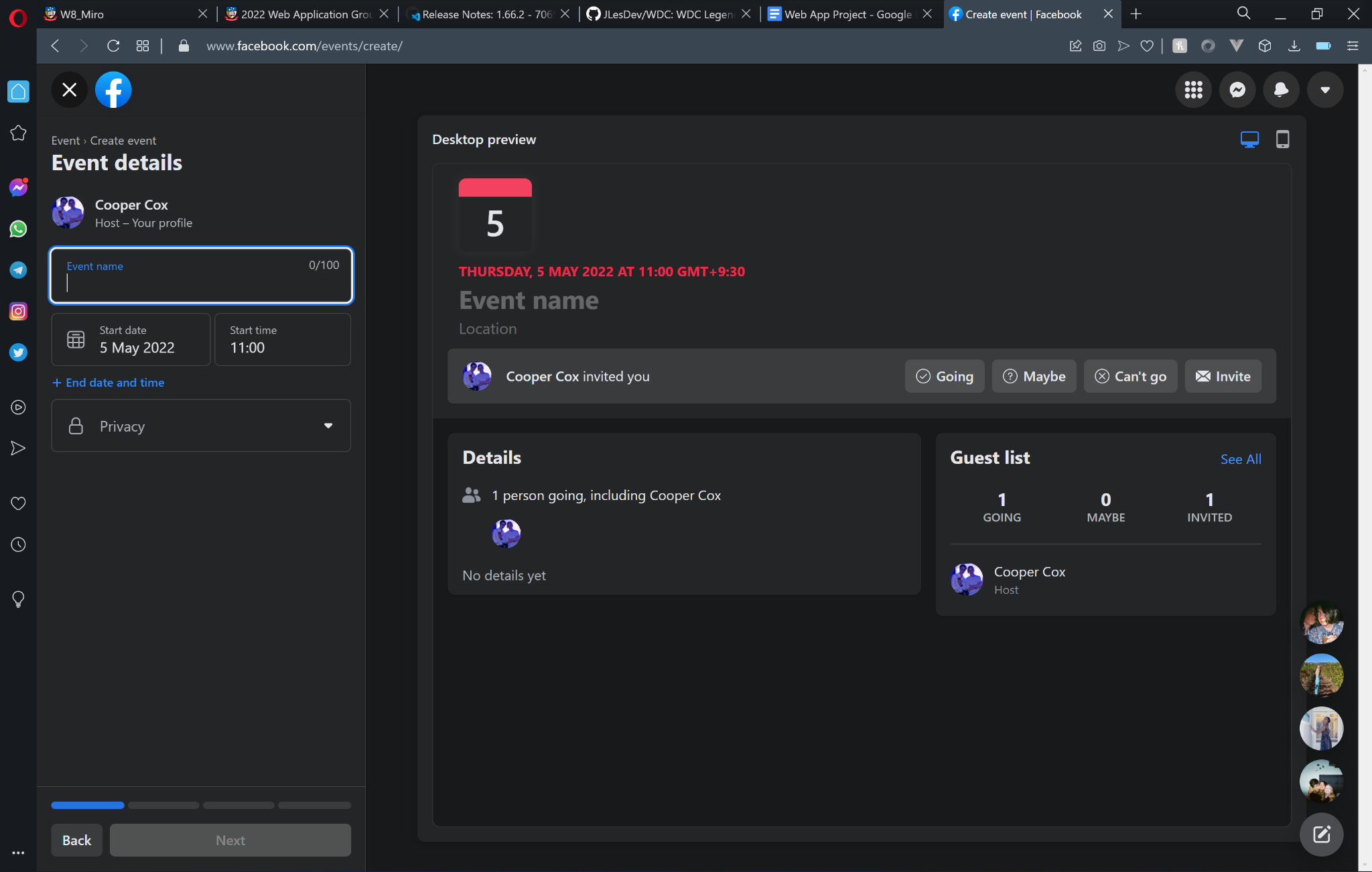
* Users should be able to sign up and log in to the system in order to  
  + Sign up/log in.
  + Manage their user information. - including deletion of account, changing email, changing email notification settings
  + Create a new event.
  + Generate a link for people without accounts.
  + Specify their availability for an event.
  + Link their calendar to automatically check their availability.
  + See times when everyone is available for an event.
  + Confirm/finalise an event time.
  + Add the finalised event to their calendar.
* System Admins should be able to log in in order to:
  + Manage their user information.
  + Manage Users.
  + Manage Events.
  + Sign-up other Admins.
* Users should be able to choose to link a social media/email/other account, allowing login via that platform, to make logging in easier.
* Users without accounts should be able to specify their availability for an event provided by a custom link.

One (1) of the following three (3) special features must also be present (2 for PG Students):

* Email notifications
  + Users can request that email notifications be sent to them that include updates when users responed, availability is confirmed, the event is finalised, or the event is cancelled.
  + Users should be able to choose which types of email notifications are sent

### Research

Think about and explore web pages & apps. What might a social event planning system look like?

* Facebook is a good example of an efficient and clean planning system with simple options such as event name, start date and time, inputs labelled on the left and a preview of what the event would look like to those invited on the right.
* 
* Stuff you can interact with on edge of screen (left or right) for fitts law
* Stuff you can’t interact with but need to see in the middle (event name, time etc)

**What information might users be required to supply?**

* Best to keep the information we receive from the users as minimal as possible so that users feel more comfortable with our service. We can do this by only asking for essential information.
* Name - To know who it is
* Email - for notifications and account creation
* Username - To identify each user
* Password (if making account for our platform instead of using google/facebook login)

**How will they log in?**

Users will have the ability to log in using:

* Accounts created for *Planiac*.
* Facebook log in
* Google Account
* Permissions to access Google Calendar itself.
* Sources: <https://developers.facebook.com/docs/facebook-login/web> <https://www.youtube.com/watch?v=XaPm7p2j_mM>

Users also do have to have an account to use our service and can receive a custom invite link to events.

**How will users specify availability?**

Users will be able to respond to an event on whether they are going or not going similar to facebook style events. Event creators will be able to see if there are any clashes on any invitee’s google calendars assuming the invitee has this setting on.

**How will their calendar integrate?**

* Users will be able to see their google calendars on the homepage with events they have been invited to automatically added to it. Event creators will be able to see if there is a clash with the time of their event and any of the invitees Google Calendars.
* Source: https://www.youtube.com/watch?v=vksO63XQQ3w

**You are encouraged to think about additional features for this application; we have outlined the minimum.**

* Research existing websites/apps to get ideas for your web application.
  + Record any pages/services you find that are good examples of what your site could look like. What parts work well. What parts don’t work well?
    - * Facebook is a great example of a clean and user friendly event planning system with simple UI and instant preview of what you are creating. Only downside might be how it is slightly tedious and requires lots of steps to confirm your entries, ours could be simpler and just a one step confirmation from the user to move on and create the event.
  + Also record any examples you find of features you might want to include in your site.
    - Email notifications if your facebook friends responded to an event. Email notifications/reminders of when your event is happening.

### Basic Design

Once you decide what a user will be able to do, then you need to think about how they will do each action and how you will make those actions visible on your site – keeping in mind the articles and lectures on design and usability.

* Draw/Sketch (don’t code) a basic design for what your site will look like.
  + It should be obvious from the drawing what each part of the site does.
  + Be sure to include any dimensions for the layout of the site and its components:
    - Are components fixed in size or do they stretch?

Dynamic sizing to adapt to different display sizes or mobile device

* + - If the window is resized, how do elements change?
      * + Use relative sizing
  + How might the site look on a phone vs desktop?
    - * Phone will be portrait, desktop will be landscape
* Thinking about the features in the description section, start to plan how each of these features of your site will function, plus any additional features that you might like to have.
  + Where does this feature appear and how does the user interact with it?
  + What happens when a user performs one of these interactions?
  + If a change occurs to the page/site following an interaction, clarify what that will be and draw if necessary.

# Describe each feature.

* + - **When a user performs an action on signup/log in**: the signup and login page will be its own html document where if the user is not logged in on opening the home page, a 4xx status update will be sent to automatically redirect to the login signup page. From there the user can then select to login or signup, as well as having options to login with facebook or google. if the user clicks login, they will enter their username and password and the program will check if the user exists and if they do, they will be redirected to the main page. If signup is clicked, the user will enter a username, password and email and systel will check if username is free, if so creates and account, logs user in and redirects to main page. If signing in with google or facebook, assigns unique id (if new user) and proceeds to redirect to main page and log user in. depending on whether login or sign up is clicked, an additional input box will appear for inputting an email. Then as user logs in, redirection to main page will occur.
    - **When a user performs an action on managing their user information:** a settings page will be created for the user to manage their account, accessed by a tab on the navbar for settings which will direct them there. The user can then change their email address, notification settings or delete their account. User will input new email in an input box and click confirm where their email will then be changed in our database, if they want to update notification settings a dropdown is provided to select email notification preferences/frequency, changing on dropdown item selection. There is also the option to link or unlink facebook or google account in the settings which will redirect them to the fb/google login screen then back to the settings page after. Then finally, a red deletion button will on onclick, confirm with the user if they are sure about their selection and if so, proceed to delete their details from the database and system, then log them out.
    - **When a user creates a new event:** The user will be directed to a unique page when they are creating an event where a series of fields asking for details on the event will appear on the left hand side of the screen. A preview of what the actual event page will appear dynamically on the rest of the rest of the screen allowing the event creator to have an idea of what they're creating, similar to how it works on facebook.
    - **When a user generates a link for a new account:** The event organiser can generate a unique link to send to people that do not have an account on the website to invite them to their event. When a user follows the unique sent to them by the event organiser they will instantly be asked to input a username for the event organiser and other attendees to identify who they are. No other account details will be required as the user who follows the unique link will not actually have to create an account.
    - **When a user specifies their availability for an event:** They are shown their google calendar and prompted whether to accept the event or not. If they do accept the event it adds it to their google calendar and shows that they are attending on the event page. If not, the event is declined. The fact that they declined the event is shown on the events page. They can change their option at anytime before the event starts.
    - **See times when everyone is available for an event**: This will appear when the event planner is inviting people to the event, then we can scan the calendars of users invited to check if they don't have anything planned on that date, a warning symbol will be displayed next to the names of those not free to indicate that they already have something on. - Could even just do an internal planning system, checking if your friend has any other events going on simultaneously from the website.
    - **System admins should be able to login in order to manage their user information, manage users, manage events, sign up other admins**: if the user is an admin, there will be an extra button on the settings page to take them to an advanced settings page that only admins can access, this will allow them to manage users (being able to remove users if violating terms, change their details etc), then they can modify users events as well as select users from existing users to become an admin where that user will then be given admin privileges.

**Once you have a basic design, you now need to review and tweak it for usability and accessibility.**

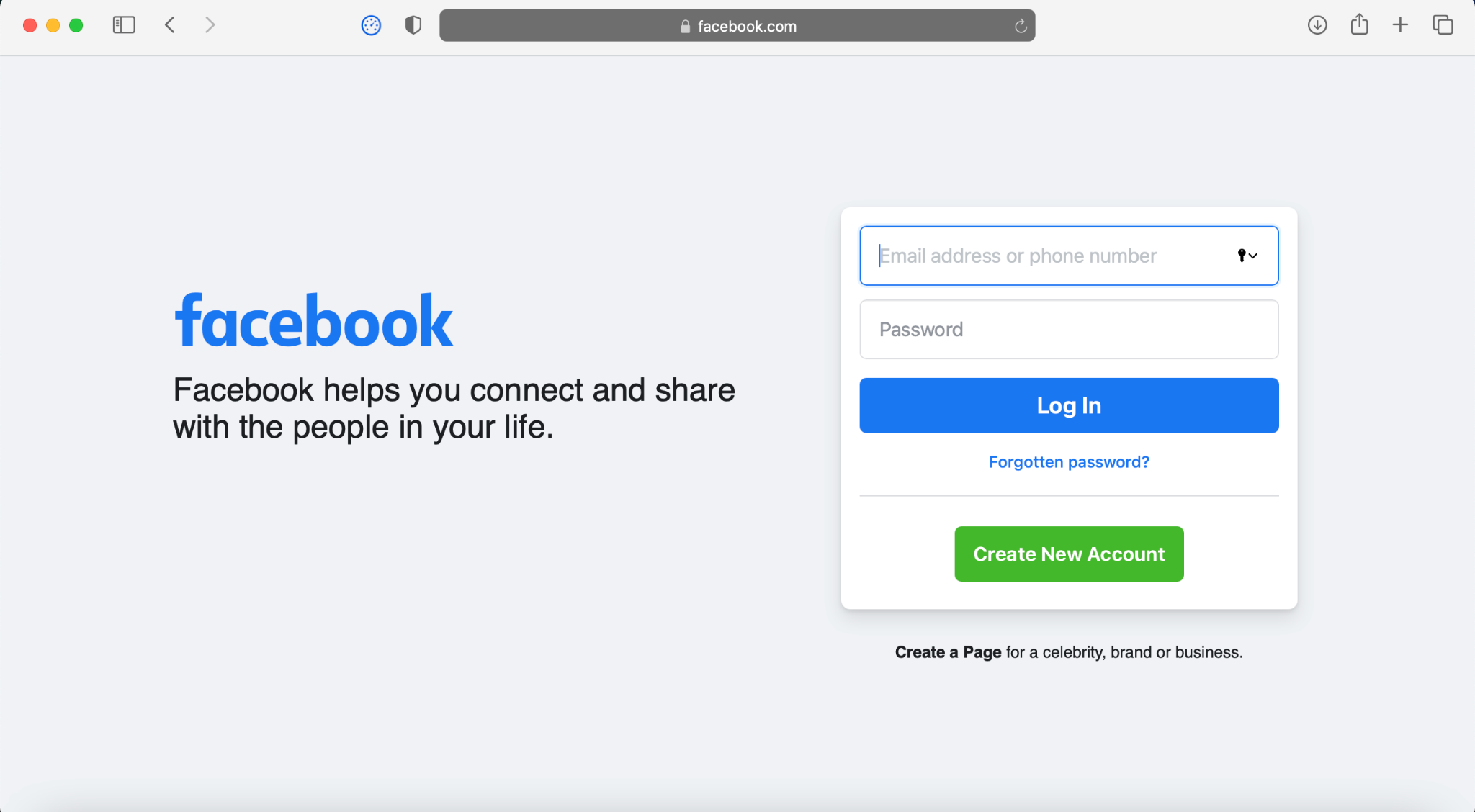
* **Review your design.**
  + How does it minimise kinematic and cognitive load?
    - * Stuff you can interact with on edge of screen (left or right) for fitts law.
      * Stuff you can’t interact with but need to see in the middle (event name, time etc)
      * Important features always available on the screen (log in/home/create event). All of these features at the top edge of the screen within the banner.
  + Does it meet the standards & heuristics referenced in lectures and on MyUni?
    - * Use icons that are self-explanatory
      * Item names explain themselves - e.g. “Create Event” “Delete Event”
      * Validation and linting all done, this is a priority. Nothing will be submitted if it doesn’t validate.
* Have another group review your website design.
  + Which parts of your site were clear. Which weren't?
  + According to the other group our plan of what we needed was very clear with the different pages and how they interact with each other clearly documented. It was not very clear how our calendar will interact with the different parts of our application. It also was not very clear what the admin page will do.
  + Record the feedback from the other group and any findings.
  + The other group suggested making our settings page less cluttered and separated into different pages which can be rendered at different times using Vue.
* Record any updates to your design made as a result of the above review.
* We changed our settings page to load different sections of settings using buttons found on the left hand side of the screen triggering vue if conditions. We also edited our explanation of our calendars implementation into our program and a section explaining the admin page and what admins can do.

**Data Plan - Cooper**

**{{ insert data plan here }}**

**Pages:**

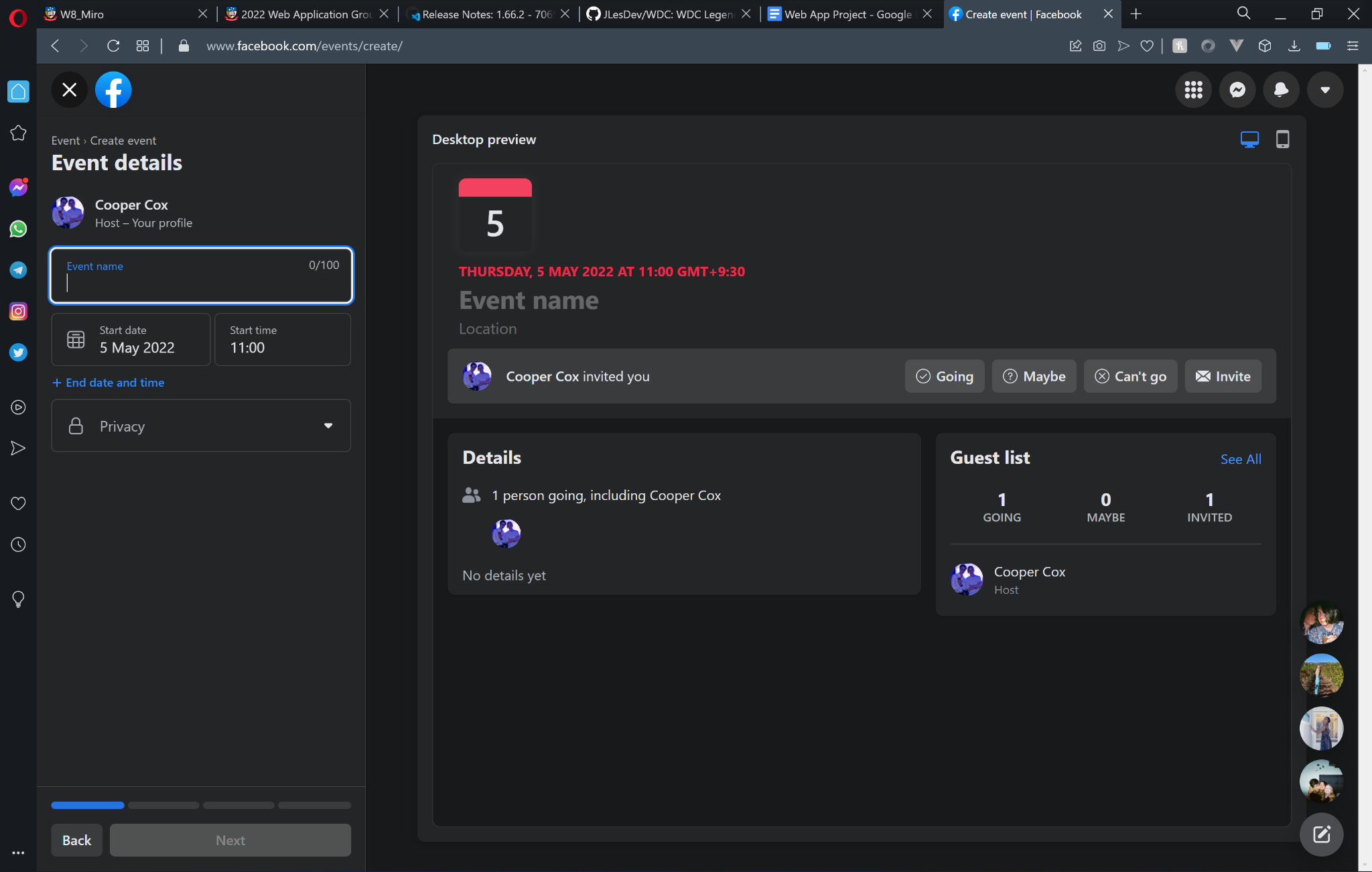
**Home page for when you’re not logged in**

e.g.

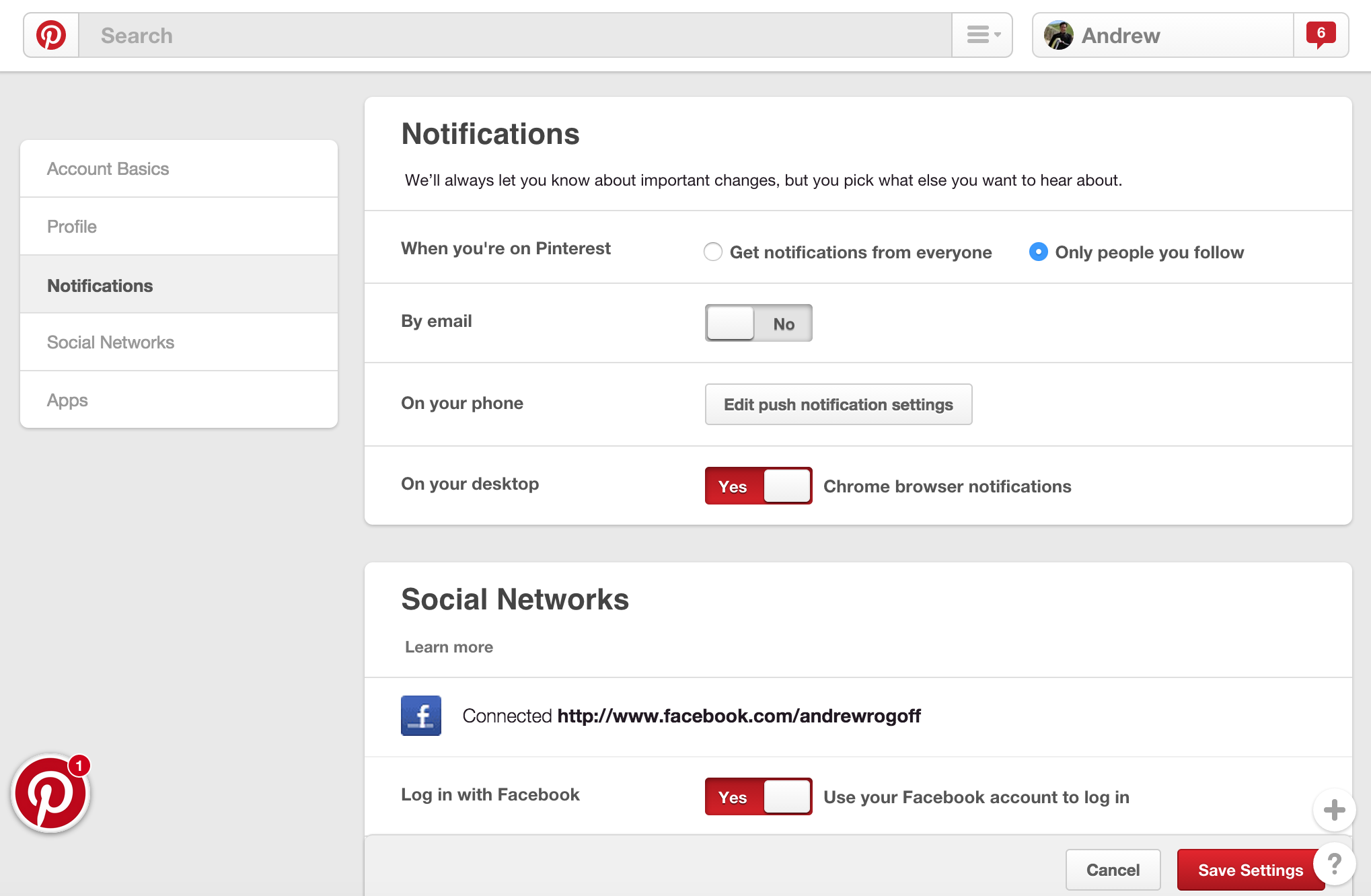
**Log in/Sign up page**

**Event page (accessible when you are logged in)/Upcoming Events (Home)**

**Create Event**

* 
* E.g. facebook event creation

**User Settings Page**

E.g. Youtube settings page

**Admin page**

### Milestone 1 Planning - Building your Web App

Now that you have a solid design, it’s time to start coding it! You should now have a firm grasp on these languages and be able to implement ALL of the static elements of your website.

* Use HTML and CSS to implement all of the parts of your site.
* Follow best practices as outlined in lectures and exercises.
  + Ensure CSS is used appropriately to provide a consistent and maintainable style across all your pages.
  + Ensure your implementation passes validation.

As you build your website, begin to add interactive features using JavaScript and Vue.

* Use JavaScript and Vue to further build your design.
* Start to add client-side code for users / venues / health-officials to interact with the system  
  + Use test data/variables to simulate client-server interactions as needed and to help you test your code.
  + Start to build your calendar features with one of the Calendar APIs shown in lectures.
* Follow best practices as outlined in lectures and exercises.
  + Ensure JavaScript files are used appropriately to provide a consistent and maintainable code across all relevant pages.
  + Ensure your implementation still passes validation after modifications.
  + Ensure your JavaScript and Vue passes linting.

Do not worry about making user interactions persistent (i.e. user changes remain after a refresh) at this stage, but do begin to think and make notes about which interactions will need server resources.

# Planning your Routes and Database

Now that you have a solid design and basic client side implementation, you can start thinking about the content that your web application will be dealing with, and how.

* For each of the features in your website, create a data plan that lists the different pieces of content/information that your web application will be dealing with & determine where it should be stored, what format, and where it should be processed.
  + Where does the information come from?

The information comes from their login details or from integration through their google or facebook account.

Information about events comes from information event hosts fill in to create an event.

* + What form should it take?

SQL database.

* + If the information is on the server, what will the client need to send to retrieve that data?
  + If the information is on the client, how will it be sent to the server?
  + Does all information need to be stored on the server?
  + What processing needs to be done to make the data useful?

Once you have an idea of the different pieces of data that you'll need to be working with, you can start thinking about how you'll store that data on your server.

* Using the methods outlined in lectures and pracs, create a Database Schema diagram for your web application's data.
* {{ insert database scheme here }}

# Research Sources

## See others' calendars side by side

You can see guests' calendars side by side in Day view when you create an event. Make sure you have turned on this ability by going to Settings Settingsand then Settings and then View options.

1. On your computer, open [Google Calendar](https://www.google.com/calendar).
2. On the left click Search for people.
3. Start typing someone’s name and choose the person you want to meet with.
4. At the top left, click Create Plusand then Event.
5. Click Find a time. If the guest you invited has shared their calendar with you, you'll see their calendar alongside yours. Guests marked as optional won’t show up.

Note: To check if a room is available, click Add rooms or Add location. Then, choose a room.

Facebook code:

FB.logout(function(response) {

// Person is now logged out

});

FB.login(function(response) {

if (response.status === 'connected') {

// Logged into your webpage and Facebook.

} else {

// The person is not logged into your webpage or we are unable to tell.

}

});

FB.login(function(response) {

// handle the response

}, {scope: 'public\_profile,email'});

FB.login(function(response){

// handle the response

});

FB.getLoginStatus(function(response) {

statusChangeCallback(response);

});

Full Facebook Login Example

<!DOCTYPE html>

<html>

<head>

<title>Facebook Login JavaScript Example</title>

<meta charset="UTF-8">

</head>

<body>

<script>

function statusChangeCallback(response) { // Called with the results from FB.getLoginStatus().

console.log('statusChangeCallback');

console.log(response); // The current login status of the person.

if (response.status === 'connected') { // Logged into your webpage and Facebook.

testAPI();

} else { // Not logged into your webpage or we are unable to tell.

document.getElementById('status').innerHTML = 'Please log ' +

'into this webpage.';

}

}

function checkLoginState() { // Called when a person is finished with the Login Button.

FB.getLoginStatus(function(response) { // See the onlogin handler

statusChangeCallback(response);

});

}

window.fbAsyncInit = function() {

FB.init({

appId : '{app-id}',

cookie : true, // Enable cookies to allow the server to access the session.

xfbml : true, // Parse social plugins on this webpage.

version : '{api-version}' // Use this Graph API version for this call.

});

FB.getLoginStatus(function(response) { // Called after the JS SDK has been initialized.

statusChangeCallback(response); // Returns the login status.

});

};

function testAPI() { // Testing Graph API after login. See statusChangeCallback() for when this call is made.

console.log('Welcome! Fetching your information.... ');

FB.api('/me', function(response) {

console.log('Successful login for: ' + response.name);

document.getElementById('status').innerHTML =

'Thanks for logging in, ' + response.name + '!';

});

}

</script>

<!-- The JS SDK Login Button -->

<fb:login-button scope="public\_profile,email" onlogin="checkLoginState();">

</fb:login-button>

<div id="status">

</div>

<!-- Load the JS SDK asynchronously -->

<script async defer crossorigin="anonymous" src="https://connect.facebook.net/en\_US/sdk.js"></script>

</body>

</html>