Group 605: PRESENTATION AREAS

 Justin, Christa, Omar, Mohamed

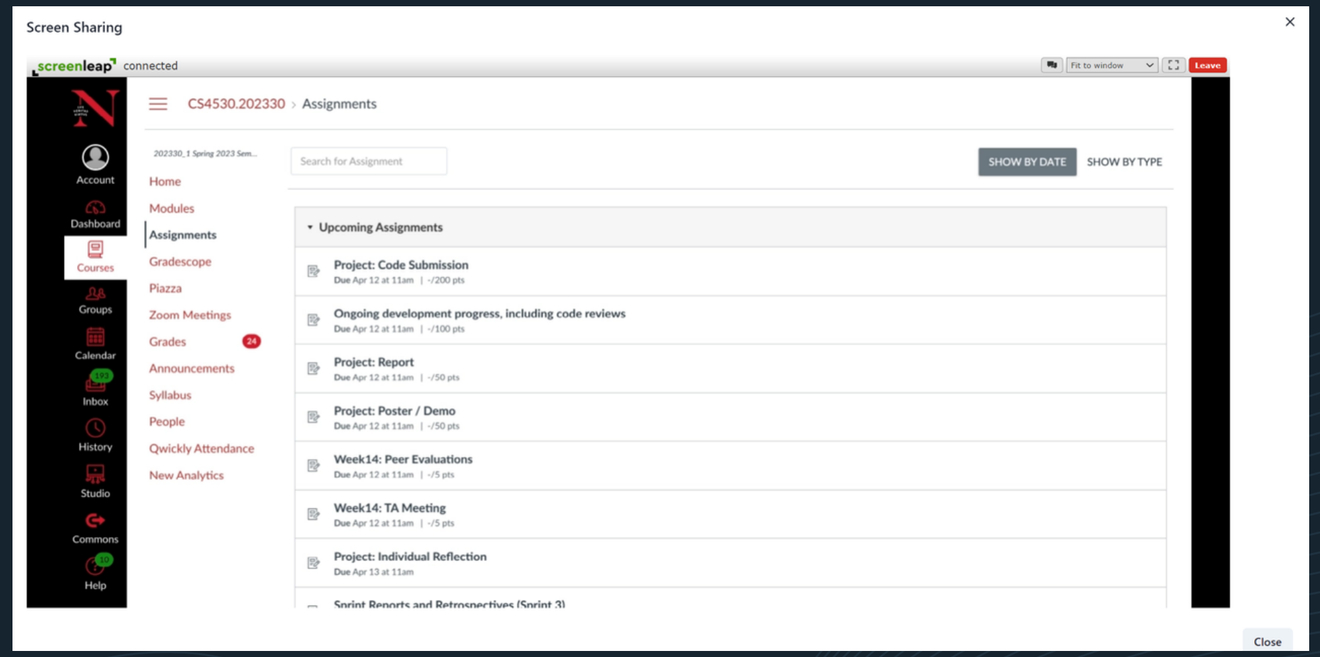
Why a Presentation Area?

 A presentation area allows for real-time ***dynamic collaboration*** and flexibility to share more ***complex and visual information*** with each other

Presentation Area Features

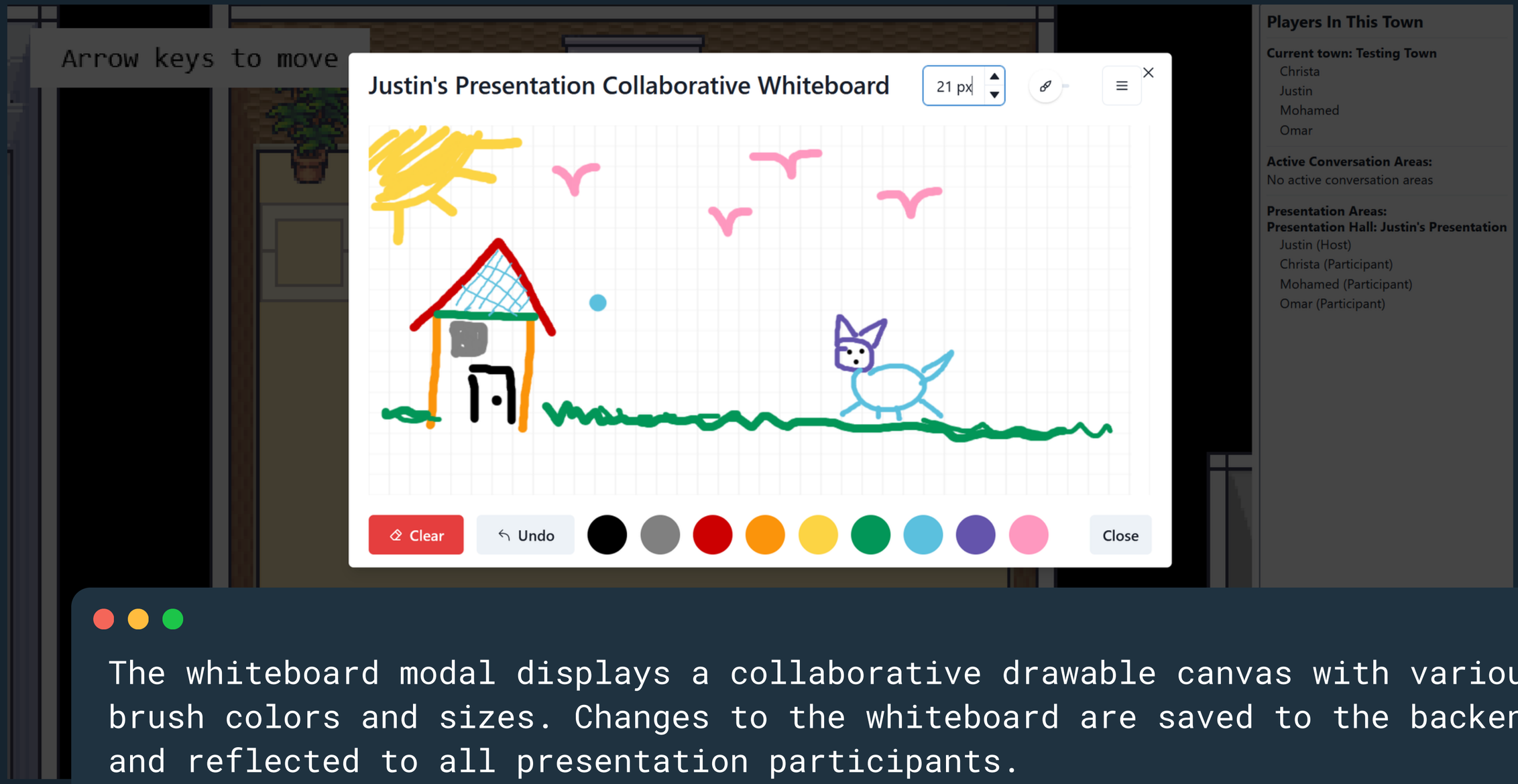
Screensharing

- Only host can start the screensharing feature
- Host generates a link from Screenleap.com, link used when creating the area
- Participants view host's screen broadcasted via Screenleap's API



The Screenshare modal powered by Screenleap's API

Collaborative Whiteboard



The whiteboard modal displays a collaborative drawable canvas with various brush colors and sizes. Changes to the whiteboard are saved to the backend and reflected to all presentation participants.

Tech Stack & Design

- PresentationArea is an extension of the Interactable area interface
- PresentationArea represented as an 'object' in tilemap, each constructed + rendered through Phaser
- All frontend components rendered + updated through Chakra UI and React Hooks. The presentation area's UI handled via:
  - Screenleap API (screenshare)
  - React Canvas Draw API (whiteboard)
  - LZ-String API (string compression for backend)
  - Chakra UI (modal to display features above and other general UI features)
- Updates to the presentation trigger calls to the backend to sync changes to all clients. Backend emits interactable updates to listeners in the frontend, which re-renders for all clients via hooks.

Major Challenges

Screenshare

- Attempt to use native browser screensharing API to sync with existing Twilio API
- Difficult to extend Twilio code, time consuming
- Quick switch to backup plan: integrate screenshare with the help of third-party API (Screenleap)

Whiteboard

How to make a 'live' whiteboard where changes are seen across every client?

1. mouseup() saves canvas state in drawing string
2. Compress drawing string, send to backend
3. Backend interactable update -> clients, clients decompress and load canvas state

Changes in Scope

- Setbacks took up time, especially switching to backup plans
- Forgo couple optional/desirable features such as integrated chat and emote reactions, but add some QoL such as social bar for presentations

Team Member Contributions

Backend Interface Design

Justin

Frontend Interface Design

Christa

UI Design

Justin

Screensharing Frontend + Backend

Omar & Mohamed

Whiteboard Frontend + Backend

Justin & Christa

Testing

Everyone

Live Demonstration

https://spring-23-team-605.netlify.app/

THANK YOU!

Please ask if you have any questions!

CREDITS.

Presentation Template: SlidesMania
Fonts used in this presentation: Roboto Mono and **Roboto Bold**

CS4530 Final Project: Presentation Areas

Group 605: Justin, Christa, Omar, Mohamed

About Our Feature

Covey.Town tailors to a professional audience. We notice that there is room for improvement when interacting with others in a professional and social environment, and having only a camera and microphone may not be sufficient. Thus, it makes sense if there is some method for presenting, such that a speaker can have a visual aid when hosting a meeting.

Our group developed a new feature that gives clients access to additional tools when interacting with others: **Presentation Areas**. Presentation Areas are specific interactable areas on the map that users can enter. The first user to interact with the presentation area is the host; subsequent users who enter are participants.

Hosts may start *screensharing* to participants. In addition, all members in a presentation area may draw on a *collaborative whiteboard canvas*. The addition of a presentation area allows for real-time dynamic collaboration and flexibility to share more complex and visual information with each other.

Demo and Source

Our public demo site is available at: https://spring-23-team-605.netlify.app/

The source repository is available at: https://github.com/neu-cs4530/spring-23-team-605

An example of a presentation being held in the Presentation Hall. Members of the presentation can see the title and member roles (host or participant) on the social bar to the right.

Technology Stack & Design

We implemented the presentation area as an extension to the existing interactable interface. A presentation area is represented as an 'object' in the town's tilemap, constructed and rendered when the map loads through Phaser.

All frontend components are rendered and updated through Chakra UI elements and React hooks, along with third-party APIs from Screenleap and React Canvas Draw. Updates to the presentation title, host/participants, screenshare link, and whiteboard trigger calls to the backend to sync changes to all clients. The backend emits interactable updates to listeners in the frontend, which re-renders for all clients via hooks.

Our continuous integration pipeline runs an automated test suite on the frontend and backend components, and then deploys the site using Heroku and Netlify.

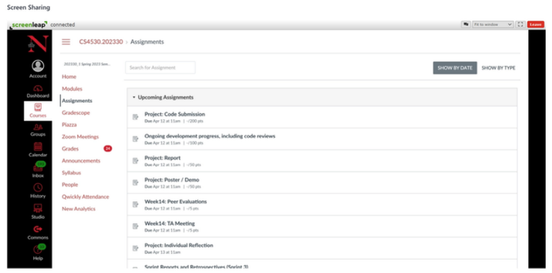
The **whiteboard** modal displays a collaborative drawable canvas with various brush colors and sizes. Changes to the whiteboard are saved to the backend and reflected to all presentation participants.

Future Work

Our current implementation contains multiple opportunities for extension and improvement. A presentation allows for a vast array of media tools beyond video, voice, screenshare, and whiteboarding. Future work on this area could include text chat, reactions via emotes, and recording.

Future work to improve existing features in the presentation area might consider a more comprehensive set of privileges for hosts to manage their presentation, in particular management of participant privileges (add, kick, allow feature, mute).

Finally, there may be native solutions to screensharing. Our group was able to create a screenshare stream without the use of a third-party API but struggled to broadcast this stream information to all clients.



The **screenshare** modal powered by Screenleap API