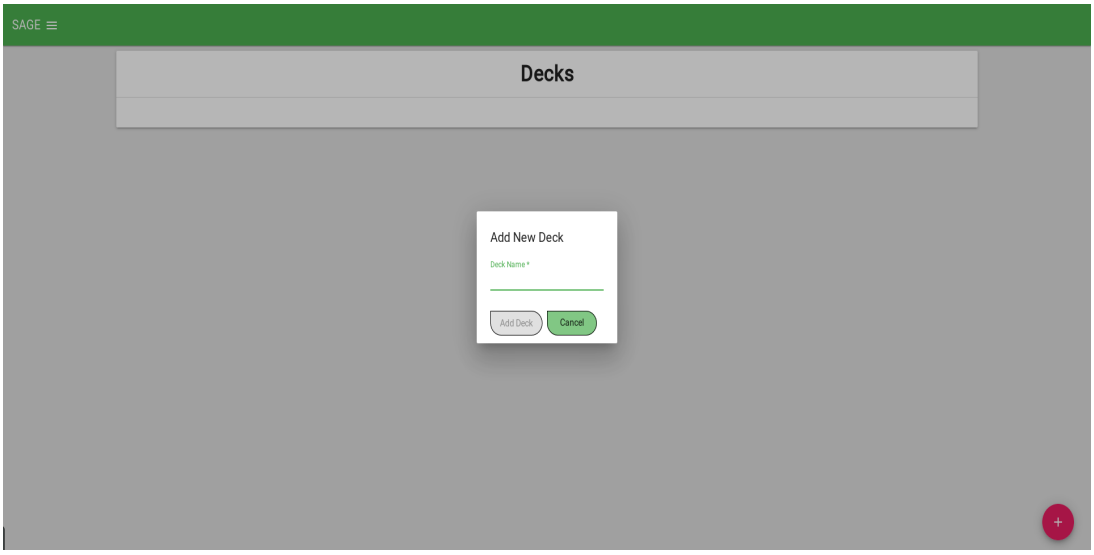
SAGE Vocabulary Game

SAGE is:

- A web-based vocabulary game
- Designed to be used in a classroom setting
- Able to be used on mobile devices
- Instructors can easily create decks and cards for students to use
- OAuth is used for teachers to identify themselves

SAGE's Home Page



Deck Creation Page

Tools and Platforms Used

- Angular - Front-end tool used to build all of the web pages
- MongoDB - Backend to store cards and decks
- Spark - Hosts the server
- IntelliJ IDEA - IDE used for the entirety of the project
- Karma - Testing tool used in the agile process
- Gradle - Automated build tool used to manage the stack
- OAuth - Google's authentication API to identify teachers
- Agile approach - The style our team used to build this project
- Zenhub - Used to manage our issues and create a burndown chart
- Github - Repository system

Agile Approach

- Group stand up meetings
- Always working together where we could easily talk
- Iterations usually lasted two weeks
- Showcases with customer at the end of each iteration
- Shopping with customer at the beginning of each iteration to decide what the next best steps are
- Good testing environment with Karma and e2e testing
- Everyone worked together on everything, no matter what they excelled at
- Burndown charts to make sure we were on track
- Issues made with story points to properly manage how long things would take

Example of my Contribution

Here is a short example of what I helped do on the front-end web development, first giving an excerpt from the HTML, and then from the Typescript. This is how we implemented editing deck names using Angular forms.

```html
<form id="name-form" #newTitleForm="ngForm">
    <mat-form-field id="title-input" [ngClass]="{'hide': this.editMode != true}">
        <input id="text-input" matInput required #input type="text"  [(ngModel)]="newDeckTitle"
               placeholder="{{deck.name}}" name="newDeckTitle" (keyup.enter)="saveEdit()" >
    </mat-form-field>
    <div class="icon-buttons">
    <button [ngClass]="{'hide': this.editMode != false}" id="edit-name" mat-icon-button class="teacher-buttons"
          id="edit" (click)="changeMode()"> <i id="edit-icon" class="material-icons">mode_edit</i>
    </button>
     <button [ngClass]="{'hide': this.editMode != false}" id="delete-deck" mat-icon-button class="teacher-buttons"
           id="delete" (click)="deleteDeck()"><i id="trash-icon" class="material-icons">delete</i>
    </button>
    </div>
    <button [ngClass]="{'hide': this.editMode != true}" color="warn" mat-raised-button class="round-button"
        id="save" (click)="saveEdit()" [disabled]="!newTitleForm.form.valid" type="submit">Save Changes
    </button>
    <button [ngClass]="{'hide': this.editMode != true}" color="warn" mat-raised-button class="round-button"
        id="cancel" (click)="cancelEdit()">Cancel
    </button>
</form>
```

```typescript
changeMode() {
    if (this.editMode == false) {
        this.editMode = true;
    } else {
        this.editMode = false;
    }
}

/*
Makes a call to the server to update the Deck with the new title of
locally as well so that the page does not need to be refreshed. Cha
 */
saveEdit() {
    this.deckService.updateName(this.newDeckTitle, this.deck._id);
    this.deck.name = this.newDeckTitle;
    this.changeMode();

}

// Opens the delete deck dialog, which handles the service request
deleteDeck() {
    this.openDeleteDeckDialog();
}

cancelEdit() {
    this.changeMode();
}
```