**Design**

**Specification**

**for**

**SmartBard**

**Requirements for Version 2.0**

| **Team:** | Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin |
| --- | --- |
| **Faculty Advisor:** | Jeff Salvage |
| **Project Stakeholder:** | Pam McGonigle |

**<May 20, 2023>**

Revision History

| Name | Date | Reason for Changes | Version |
| --- | --- | --- | --- |
| Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin | 1/26/22 | Initial draft and sections outlined. | 1.0 |
| Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin | 2/2/23 | Revised design doc. | 1.1 |
| Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin | 2/14/23 | Updated architectural diagrams and added a traceability matrix. | 1.2 |
| Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin | 4/25/23 | Added detailed architecture section and completed traceability matrix. | 1.3 |
| Peter Chong, Brandon Ly, Ricky Taing, An Tran, Sandy Yang, Jarod Durkin | 5/20/23 | Updated and finalized design requirements. | 2.0 |

Table of Contents

[**1. Introduction 5**](#_vilw2emoxr3z)

[1.1. Purpose 5](#_xm33lsb4po6e)

[1.2. Design Goals 5](#_7ocmfszado3d)

[1.3. References 6](#_o5rhfnssw2qt)

[1.4. Definitions, Acronyms, and Abbreviations 6](#_rni0t08wx03h)

[**2. System Overview 7**](#_y129gcju2i9d)

[2.1. Technologies Used 7](#_94k96cvkd847)

[2.2. Component Diagram 8](#_vzwrsdxjvlqi)

[2.3. Sequence Diagrams 9](#_vstcuh7pm1iq)

[**3. Data Model 14**](#_l8mlosdrgsts)

[3.1. Database Schema 14](#_ueweinbfgyuw)

[3.2. Announcements Table 15](#_jtd2uknxbaip)

[3.3. Audit Log Table 17](#_sxglalliu96a)

[3.4. Users Table 18](#_ksim36a4h7op)

[3.5. User Settings Table 19](#_nx7uwkufzvqk)

[**4. Detailed System Architecture 20**](#_s47rz5wpucph)

[4.1. User Interface Architecture 20](#_p0vhtgb2qsxg)

[**5. Traceability Matrix 26**](#_7tcbk63jktpz)

# Introduction

## Purpose

The purpose of this document is to specify the design specifications and software architecture for SmartBard. SmartBard is a bulletin board web application for Overbrook School of the Blind designed to help those visually impaired. This document follows the functional requirements defined in the Software Requirements Specification for SmartBard document.

This document is intended for any developer, designer, and/or tester that needs to understand the architectural design of SmartBard and its specifications.

## Design Goals

SmartBard requires many different systems to interact with each other for the application to function correctly. The two facets of efficient design of SmartBard are accessibility and modularity.

Accessibility pertains to the barriers of entry for a user with any disability to use an application. SmartBard is made for Overbrook School of the Blind and thus designed with those with a visual impairment in mind. Interactions and elements throughout the user interface can be resized, colored, or even speech-driven. Therefore, components within the user interface are loosely coupled to ensure that even the most extreme edge cases can work for all users.

Modularity is considered throughout the application lifecycle and allows for SmartBard to work in unison with different types of systems. The modular authentication interface is provided such that it can be changed without directly modifying the core of SmartBard. The database interfaces are also written with the possibility of data migrations in the future. Each external dependency is given a unified way to interact with SmartBard so that the goal of modularity can be achieved.

## References

This document references the software requirements in the Software Requirements Specification for SmartBard document.

## Definitions, Acronyms, and Abbreviations

**API -** Application Programming Interface; A way for two or more computer programs to communicate with each other.

**AWS**  - Amazon Web Services, a cloud computing platform.

**MB -** Megabyte; A measure of computer processor storage.

**Post/Announcement** - An item that would be traditionally placed on a bulletin board, such as a general announcement, notice, picture, and or document.

**S3** - AWS Simple Storage Service; An object storage and retrieval service provided by Amazon Web Services.

**URL -** Uniform Resource Locator; The address of a given unique resource on the Internet.

**Visual Impairment** - Loosely defined in vision that someone, even with correction, such as with glasses or contacts, still has trouble seeing things.

# System Overview

## Technologies Used

SmartBard is implemented in Javascript using the React framework. The application contains the react-bootstrap library to both quicken and improve development. React-bootstrap contains a variety of components that allow the placement of custom components into a grid-like system, which also provides the basic HTML DOM elements in a component form, allowing smoother integration with the React framework. The internal API for SmartBard uses Node.JS, a JavaScript runtime with the open-source library Express.JS.

Infrastructure for SmartBard is provided by AWS, which contains a managed virtual machine (VM) and PostgreSQL relational database system. The user interface is delivered to the user through the content delivery systems AWS Simple Storage Service (S3), and AWS CloudFront. Logging is provided with AWS X-Ray and user authentication is structured by AWS Cognito.

## Component Diagram

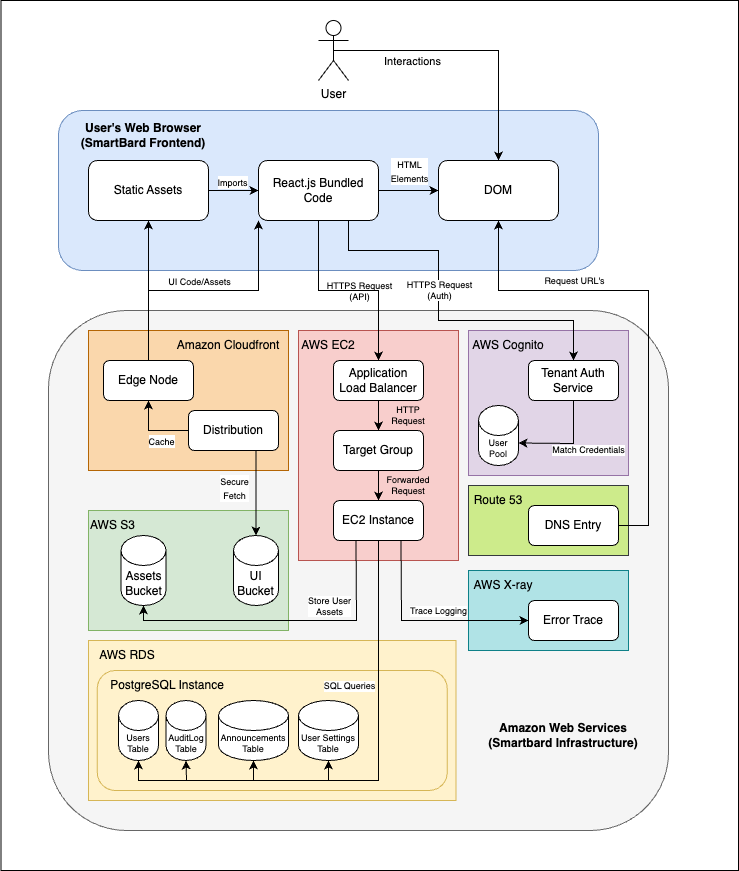


Figure 1. **Component Diagram of SmartBard**

Figure 1 shows the high-level interactions between all systems that allow SmartBard to function. The user’s web browser handles the client view of the SmartBard application and allows the user to interact with the application. The web browser interacts with several external API’s that are hosted inside of AWS. These public-facing components send various commands to other internal components inside of AWS such as the relational database service.

## Sequence Diagrams

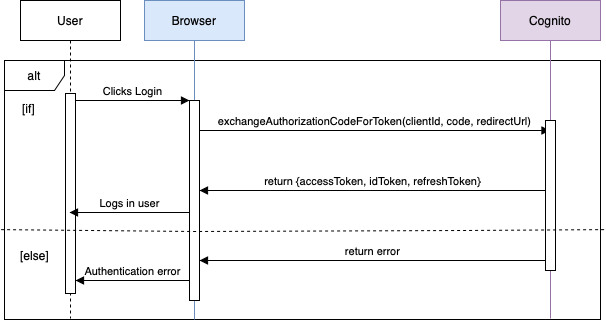


Figure 2. **Sequence Diagram for Logging into SmartBard**

Figure 2 shows the process of a user logging into Smartbard. Upon the user signing into Google, the login credentials are sent to Cognito to authenticate the user. If the credentials are valid, it logs in the user to Smartbard, and if the credentials are invalid, it returns an authentication error to the user.

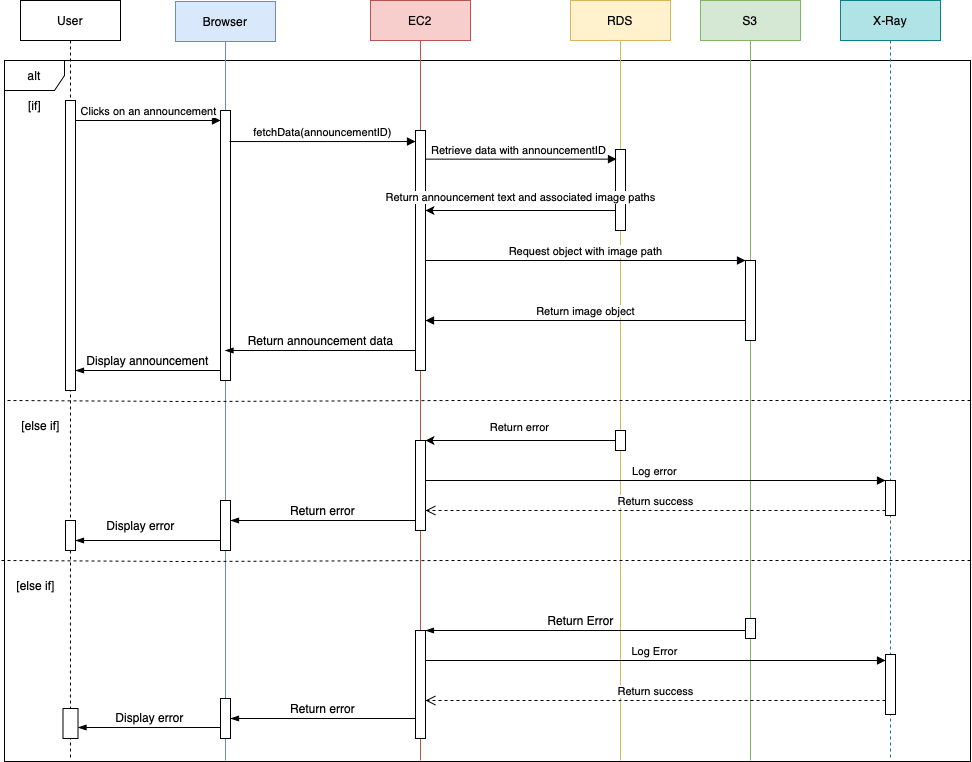
Figure 3. **Sequence Diagram for Viewing a Post**

Figure 3 shows the process of a user viewing an announcement post. The announcementID is sent to RDS to fetch the announcement data. If the post has any images, the image path is sent to S3 to retrieve any image objects. Then, it returns the complete announcement post back to the user. If there are any errors with RDS or S3, the error is logged in X-Ray, and the error is sent to the user.

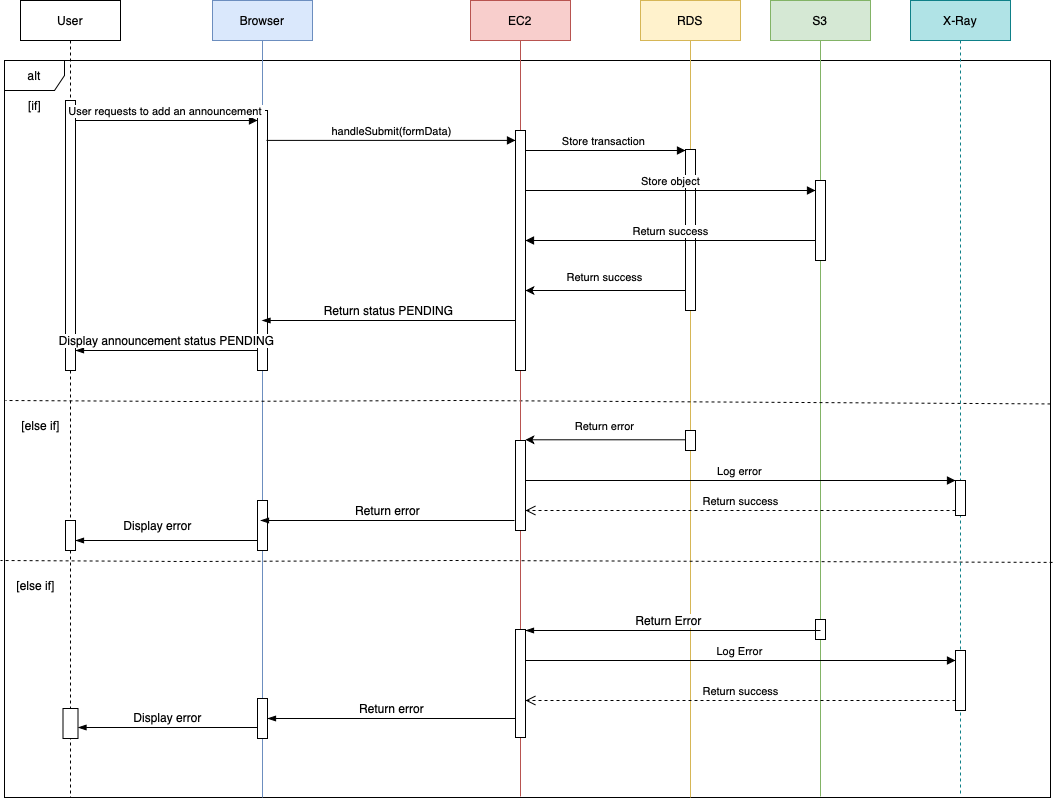
Figure 4. **Sequence Diagram for Adding a Post**

Figure 4 shows the process of a user requesting to add an announcement post. Upon filling out a new request, the request’s announcementID and associated information is sent to RDS and S3 to store appropriate information. Then, it returns to the user that their request is “Pending”. If there are any errors with RDS or S3, the error is logged in AWS X-Ray, and the error is sent to the user.

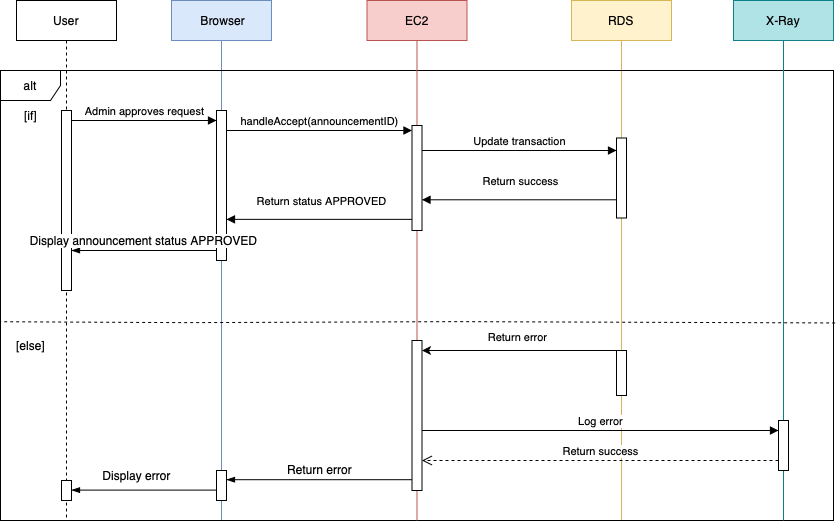
Figure 5. Sequence diagram for approving a request

Figure 5 shows the process of an Administrator approving a request. After reviewing a request and clicking on the ***Accept*** Button, the request data is updated on RDS using the announcementID, changing the status of the request, and returning back to the user that it is now “Approved”. If there are any errors with RDS or S3, the error is logged in X-Ray, and the error is sent to the user.

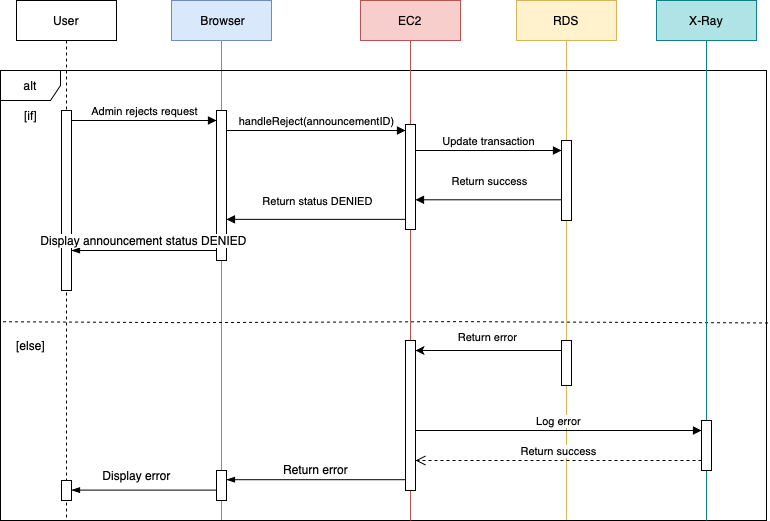
Figure 6. **Sequence Diagram for Rejecting a Request**

Figure 6 shows the process of an Administrator rejecting a request. Similar to the previous sequence for approving a request, the request data is updated on RDS using the announcementID, changing the status of the request, and returning it back to the user as “Denied.” If there are any errors with RDS or S3, the error is logged in X-Ray, and the error is sent to the user.

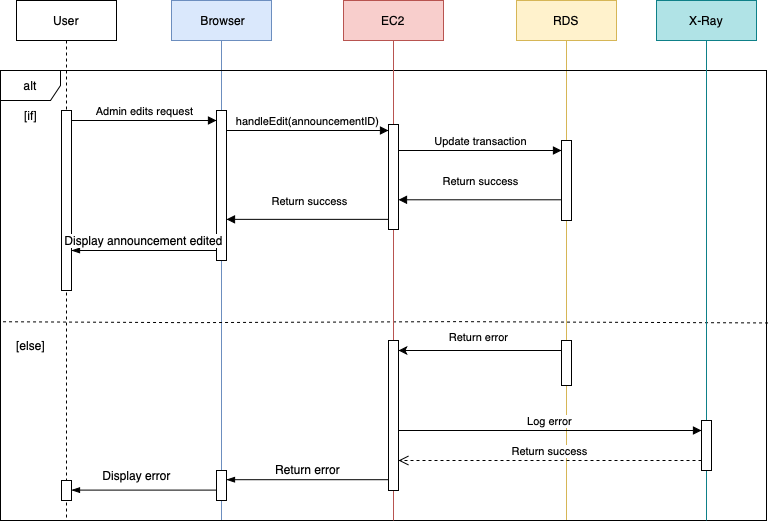
Figure 7. **Sequence Diagram for Editing a Request**

Figure 7 shows the process of an Administrator editing a request. After editing an announcement, the request data is updated on RDS using the announcementID. If there are any errors with RDS, the error is logged in X-Ray, and the error is sent to the user.

# Data Model

## Database Schema

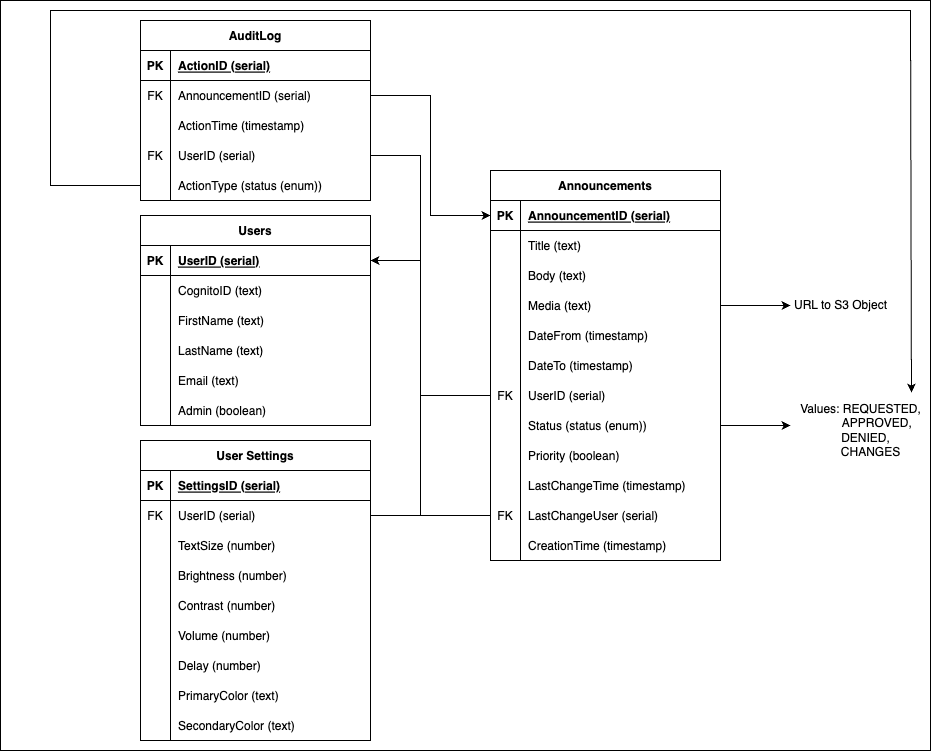


Figure 8. **PostgreSQL Database Schema**

SmartBard has a light data model consisting of four database tables, defined with PostgreSQL syntax and types. The announcement table is the primary table, with a single announcement as each row. The audit log table keeps track of all user actions throughout the application and the time at which they happened. The users table contains metadata about users that exist in the system. The user settings table persists accessibility settings between devices for each user.

## Announcements Table

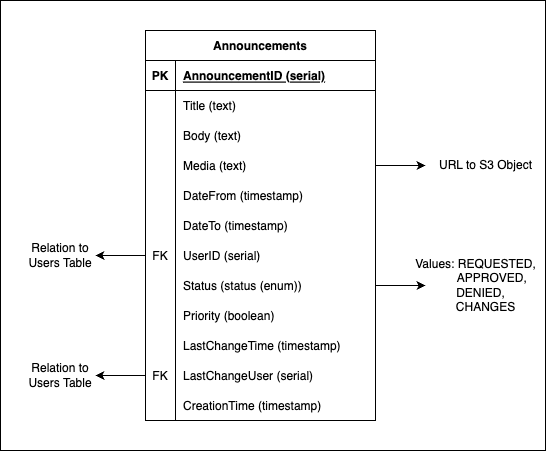


Figure 9. Announcements Table Data Model

**AnnouncementID -** Internal unique identifier for a single announcement. Primary key for this table and auto-incremented as each announcement entry is created.

**Title -** Text-based field that indicates the name of an announcement.

**Body -** Text-based field that stores the actual content of an announcement

**Media -** URL that is used for retrieving documents or multimedia that a user attaches to an announcement. This URL typically will point to an object in an AWS S3 Bucket.

**DateFrom -** Date formatted field that indicates the start date and time that the given announcement should be displayed.

**DateTo -** Date formatted field that indicates the end date and time that the given announcement should be removed from view.

**UserID -** A foreign key relation to a single row in the users table that will indicate the user that created this specific announcement.

**Status -** A custom enumerable type that indicates if the announcement is awaiting approval (REQUESTED), or if the announcement has been approved (APPROVED), denied (DENIED), or changes have been requested to the content of the announcement (CHANGES).

**Priority -** A boolean field such that if true indicates that this announcement is a priority over all others and should be displayed first.

**LastChangeTime -** Date formatted field that indicates when this announcement was last modified by any user.

**LastChangeUser -** A foreign key relation to a single row in the users table that will indicate the user that performed the last modification to the announcement.

**CreationTime -** Date formatted field that indicates when exactly this announcement was first requested.

## Audit Log Table

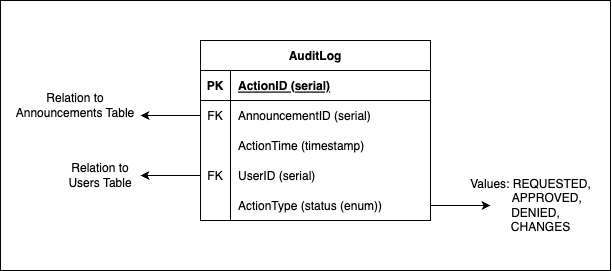


Figure 10. Audit Log Table Data Model

**ActionID -** Identifier for this particular action by a user. Primary key and auto increments as each action is performed and placed in the database.

**AnnouncementID -** Foreign key relation to the announcements table which indicates what particular announcement that this action was performed on.

**ActionTime -** Date formatted field that indicates when this action was performed.

**UserID -** Foreign key relation to users table which indicates the user that performed this specific action.

**ActionType -** A custom enumerable type that indicates what type of action was performed. If the announcement was requested (REQUESTED), or if the announcement was approved (APPROVED), denied (DENIED), or changes have been requested to the content of the announcement (CHANGES).

## Users Table

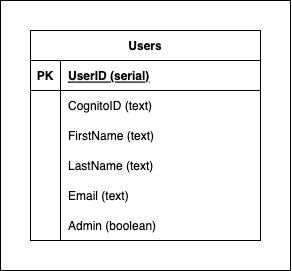


Figure 11. User Table Data Model

**UserID -** Unique internal identifier for a user. Indicates who posted announcements and performed certain actions. An auto-incrementing primary key.

**CognitoID -** Text field that relates this user to a particular user in the AWS Cognito user pool. Allows the system to verify a specific user when using external authentication.

**FirstName -** Text field that indicates a user’s first name.

**LastName -** Text field that indicates a user’s last name.

**Email -** Text field that indicates a user’s email. Users will use their email as a username when logging into the system.

**Admin -** A boolean field that indicates if this particular user has administrative privileges.

## User Settings Table

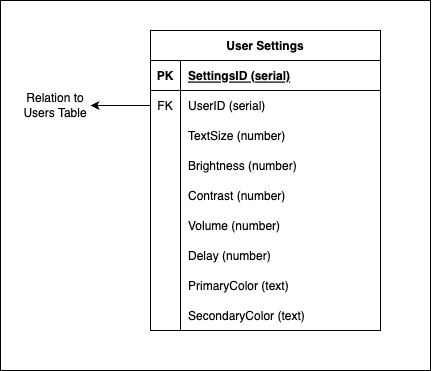


Figure 12. User Settings Table Data Model

**SettingsID -** An identifier for a particular set of settings. A primary key that auto-increments when new sets of settings are added.

**UserID -** A foreign key relation to the users table indicating the user that these settings apply to.

**TextSize -** A numerical field that indicates the text size on the user’s screen.

**Brightness -** A numerical field that indicates the brightness of a user’s screen.

**Contrast -** A numerical field that indicates the contrast of a user’s screen.

**Volume -** A numerical field that indicates the boldness of text on a user’s screen.

**Delay -** A numerical field that indicates the amount of time (in milliseconds) between when a user performs an action and the result of that action is displayed on screen.

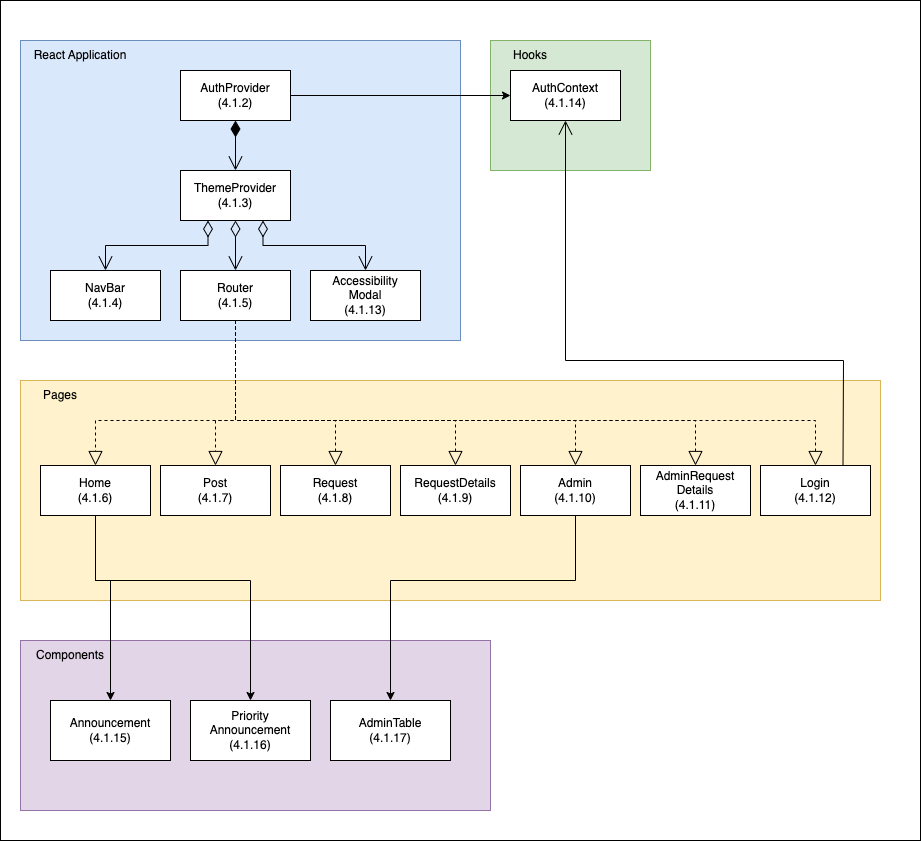
**PrimaryColor -** A text based field that establishes the primary color of user interface elements.

**SecondaryColor -** A text based field that establishes the secondary color of other user interface elements.

# Detailed System Architecture

## User Interface Architecture

* + 1. UI Class Diagram

 Figure 13. Class Diagram for the SmartBard User Interface

* + 1. AuthProvider  
       A react context used to keep track of a user’s authentication status while using the application.  
       **isAuthenticated -** A state provided to the app to determine if the user is successfully authenticated.  
       **Login -** Stores a token in the browser’s web storage to authenticate the user.  
       **Logout -** Removes the current token from browser web storage and redirects the user back to the login page.
    2. ThemeProvider

A react context that contains the different themes of the application.

**ThemeContext -** Stores the theme of the application

* + 1. NavBar

The component which is responsible for displaying the navigation bar throughout the entire application.

**Logout -** Removes the current token from the browser web storage and redirects the user back to the login page.

**Admin** - Redirects the user to the admin page.

**Request** **-** Redirects the user to the request page

* + 1. Router

A library which provides the ability to display specific components based on the url provided, which is maintained through the usage of buttons and other components.

**PrivateRouter -** Handles routing between pages.

* + 1. Home

The component that is responsible for the display and structure of what the landing page should look like.

**Announcement -** Post component of the request.

**PriorityAnnouncement -** Priority Announcement of the request.

* + 1. Post

Each Post component comprises a title, body, and image field. Each Post component also has its own associated ID which is displayed at the end of the current URL when viewing a post.

* + 1. Request

This is the component where the user is presented with a table of posts, sorted in tabs of status: Pending, Accepted, and Rejected.

* + 1. RequestDetails

Similar to the Post component, however this shows the details of the request in the format that is shown when a request is being created.

* + 1. Admin

The same as the Request component, except when the Admin is currently viewing the Request table. Admins have access to more options pertaining to posts in the Pending, Accepted, and Rejected state.

* + 1. AdminRequestDetails

This is also similar to the RequestDetails component. Admins have the option to swap posts between different states, unlike students. They can also edit posts and send them back to the pending state for the students.

* + 1. Login

Simply displays a login screen for the user to log into their account.

* + 1. AccessibilityModal

Modal popup displaying accessibility options for the application.

* + 1. AuthContext

* + 1. Announcement

Carousel of announcements for the week. Drag left or right to read other announcements.

* + 1. PriorityAnnouncement

Main announcement at the top of the page. Priority announcements can be changed by administrators.

* + 1. AdminTable

The AdminTable consists of the posts that display per tab, ready for the admin to audit.

**AdminRequestTable -** Fetches the requests submitted to the admin and formats the data into a table.

* 1. API Architecture
     1. API Class Diagram

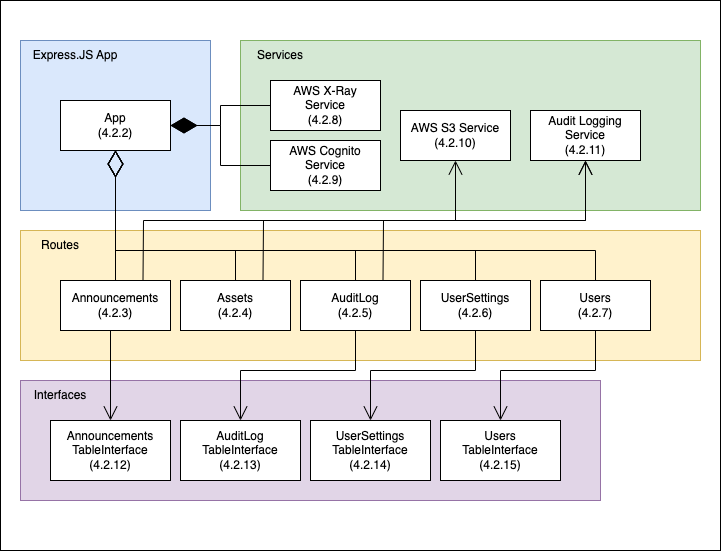


Figure 14. Class Diagram for the SmartBard API

* + 1. App  
       An instance of the Express framework for Node.JS.  
       **Use -** Establishes middleware that is used for corresponding HTTP requests to the API.  
       **Listen -** Starts an HTTP server to receive and distribute requests.
    2. Announcements  
       A router that handles all announcement data related HTTP requests.  
       **GET -** Retrieves announcements in the database, allowing for filtering based on status or date.  
       **POST -** Creates a new announcement given the data inside of the request body.  
       **PUT -** Modifies a particular announcement based on the properties provided in the request body.  
       **DELETE -** Permanently deletes an announcement from the database.
    3. Assets  
       A router that enables users to directly upload announcement-related asset data to the server.  
       **POST -** Validates and receives a file using HTTP form-data from the request.
    4. AuditLog  
       A router that allows users to retrieve the latest audit log for announcement-related operations completed by users.  
       **GET -** Retrieves the latest actions performed by users, and allows for pagination.
    5. UserSettings

A router that allows the system to retrieve custom user settings.

**GET -** Retrieves a user’s preferred settings for the application, such as font size and color theme, from the database.

* + 1. Users

A router that allows the application to retrieve user information.

**GET -** retrieves user information from the database and

verifies that the user exists.

* + 1. X-RayService

A service that traces the flow of requests through the application,

providing developers a way to visualize and analyze errors.

* + 1. CognitoService

A service that manages the login and sign-out of users to the application, integrating with Google to provide authentication.

**isValidToken -** Retrieves jwt token from request header to verify user access.

**getUserEmailFromToken -** Retrieves email from verified user token.

**initializeUserIfNotExist -** Creates a new user or update user information in the database.

* + 1. S3Service  
       A service that enables the asset route to take the files that users provide with announcements and store them inside a persistent AWS S3 bucket.  
       **uploadObjectToS3 -** Takes an asset and uploads it to an S3 bucket, returning the path to the resource.  
       **deleteS3Object -** Based on the provided resource path, delete the respective object from the assets S3 bucket.  
       **getS3Object -** Based on the provided resource path, retrieve the respective object from the assets S3 bucket.
    2. AuditLoggingService  
       A service that automatically retrieves user information and metadata about particular actions throughout the app.  
       **logAction -** Sends an entry to the audit log based on the information given.
    3. AnnouncementsTableInterface

Provides methods to interact with Announcements Table in the database.

**getAnnouncements** - Query database for all Announcements

**createAnnouncement** - Create an Announcement in the table with “vals” values passed in as a parameter.

**updateAnnouncement** - Update an existing Announcement with provided “id” argument. Change the “column” field to “newValue” passed in as parameters.

**deleteAnnouncement** - Delete an Announcement with provided “id” argument.

**getUserActivity** - Return the time when an Announcement was last changed and the person who made the change with the provided “id” argument.

* + 1. AuditLogTableInterface  
       Provides methods to interact with the Audit Log Table in the database.  
       **addLog -** Add a log entry to the audit log table based on the parameters provided.  
       **retrieveLogPage -** Gets the current audit log of actions in descending order and in paginated form.  
       **removeLogsOfAnnouncement -** Delete all logs related to an announcement, in case if an announcement needs to be permanently deleted.
    2. UserSettingsTableInterface

Provides methods to interact with User Settings Table in the database.

**getUserSettings** - Query database for all User Settings.

**getUserSettingsById** - Query database for a specific User’s User Settings by their “id” parameter.

**createUserSettings** - Create a User Setting using values from “vals” parameter.

**updateUserSettings** - Update User Settings with id “userId”. Changing the value at “column” to “newValue”.

**deleteUserSettings** - Delete User Setting with “userId” parameter.

* + 1. UsersTableInterface

Provides methods to interact with Users Table in the database.

**getUserById** - Query database for users with parameter “id”.

**getUserByEmail** - Query database for users with parameter “email”.

**createNewUser** - Create a new user in the Users Table with values passed in as the “vals” parameter.

**updateUser** - Update an existing user with values from “vals” parameter.

# Traceability Matrix

| Requirement | Description | Class Number | Class Name |
| --- | --- | --- | --- |
| **R2.1** | **Login Page** | **4.1.12** | **Login** |
| R2.1.1 | Login Button | 4.1.12 | Login |
| R2.1.2 | Continue with Google Button | 4.2.9 | CognitoService |
| **R2.2** | **Landing Page** | 4.1.6 | Home |
| R2.2.1 | Navigation Bar | 4.1.4 | NavBar |
| R2.2.2 | Request Button | 4.1.4 | NavBar |
| R2.2.3 | Admin Button | 4.1.4 | NavBar |
| R2.2.4 | Logout Button | 4.1.12 | NavBar |
| R2.2.2.1 | Click **Logout Button** to go to **Login Page** | 4.1.12 | NavBar |
| R2.2.3.1 | Click **Request Button** to go to **Request Page** | 4.1.4 | NavBar |
| R2.2.4.1 | Click **Admin Button** to go to **Admin Page** | 4.1.4 | NavBar |
| R2.2.5 | Carousel | 4.1.6 | Home |
| R2.2.5.1 | Display 3 posts, with left and right button | 4.1.15 | Announcement |
| R2.2.6 | Announcement Posts | 4.1.15 | Announcement |
| R2.2.6.1 | Priority announcement has full title and full description | 4.1.16 | PriorityAnnouncement |
| R2.2.6.2 | Priority announcement displays media attachment if available | 4.1.16 | PriorityAnnouncement |
| R2.2.6.3 | Click on any post to go to its **Detailed Announcement Page** | 4.1.15 | Announcement |
| R2.2.7 | Settings Button | 4.1.13 | AccessibilityModal |
| R2.2.7.1 | Click **Settings Button** to go to **Settings Page** | 4.1.13 | AccessibilityModal |
| **R2.3** | **Announcement Page** | 4.1.7 | Post |
| R2.3.1 | SmartBard Logo | 4.1.4 | NavBar |
| R2.3.2 | Detailed Announcement | 4.1.7 | Post |
| R2.3.2.1 | Title is centered, at the top | 4.1.7 | Post |
| R2.3.2.2 | Description is below title | 4.1.7 | Post |
| R2.3.2.3 | Picture is underneath description | 4.1.7 | Post |
| R2.3.3 | Attached Documents Button (ADB) | 4.1.7 | Post |
| R2.3.3.1 | ADB below description | 4.1.7 | Post |
| R2.3.3.2 | ADB below picture, if picture available | 4.1.7 | Post |
| R2.3.3.3 | Click on ADB to view and download documents | 4.2.10 | S3Service |
| **R2.4** | **Request Page** | 4.1.8 | Request |
| R2.4.1 | SmartBard Logo | 4.1.4 | NavBar |
| R2.4.1.1 | Click logo to go to **Landing Page** | 4.1.4 | NavBar |
| R2.4.2 | Request Table | 4.1.17 | AdminTable |
| R2.4.2.1 | Tabular format | 4.1.17 | AdminTable |
| R2.4.2.2 | Pending, Accepted, Declined tabs | 4.1.17 | AdminTable |
| R2.4.2.3 | Click each tab to display Pending, Accepted, Declined requests | 4.1.17 | AdminTable |
| R2.4.2.4 | Status and date to the right of request | 4.1.17 | AdminTable |
| R2.4.3 | New Button Table | 4.1.4 | NavBar |
| R2.4.3.1 | Click **New Button** to go to **Create Request Page** | 4.1.4 | NavBar |
| R2.4.4 | Settings Button | 4.1.13 | AccessibilityModal |
| R2.4.4.1 | Click **Settings Button** to go to **Settings Page** | 4.1.13 | AccessibilityModal |
| **R2.5** | **Create Request Page** | **4.1.9** | **RequestDetails** |
| R2.5.1 | New Request Page | 4.1.9 | RequestDetails |
| R2.5.1.1 | Title is 1 and 60 characters long with alphanumeric, space, hyphen and underscore | 4.2.3 | Announcements |
| R2.5.1.2 | Description is 1 and 280 characters with alphanumeric, space, hyphen and underscore | 4.2.3 | Announcements |
| R2.5.1.3 | Media is .pdf, .jpg, and or .jpeg file under 10 mb | 4.2.3 | Announcements |
| R2.5.1.4 | Date starts on current day or future date and ends on future date | 4.2.3 | Announcements |
| R2.5.2 | SmartBard Logo | 4.1.4 | NavBar |
| R2.5.2.1 | Click Logo to go to **Landing Page** | 4.1.4 | NavBar |
| R2.5.3 | Settings Button | 4.1.13 | AccessibilityModal |
| R2.5.3.1 | Click **Settings Button** to go to **Settings Page** | 4.1.13 | AccessibilityModal |