Juvenile Justice System

Khaled Kyle Wong
Nguyen Nguyen
Tim Nguyen
Nyan (Jonathan) Tun
Jason Wong
Alexander Chen

Project Overview

- On **May 2,** our team met with the client to discuss the key components they wanted in the application.

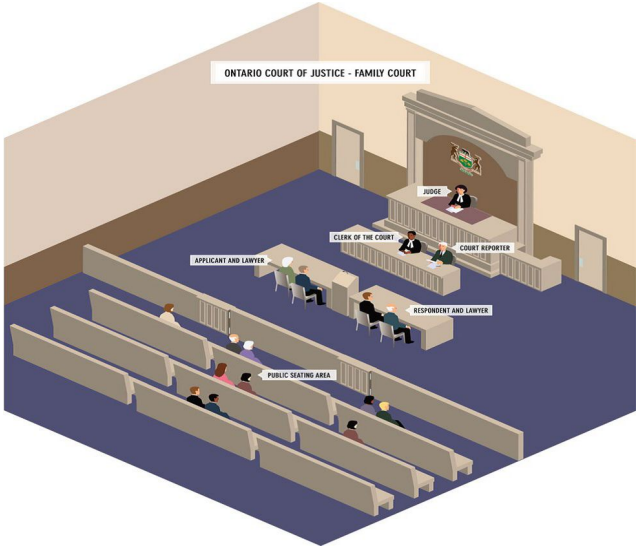
- Our clients would like us to develop an application that will teach users at a 7th grade reading level about the Los Angeles Juvenile Justice System.
- Application must run on a mobile device
- Application text must be dynamic
    - Text must be editable from an admin dashboard
    - Data will be edited by a non-technical individual

Main Components

- User facing
  - iOS application that will be used by the youths
- Admin dashboard
  - Dashboard that is private to the Justice team
  - Admins modify terms/definitions that will show up in the user facing iOS app

User App: Feature 1

- Court Roles
  - An interactive page with a picture of a court and various entities (prosecutor, defendant, judge, etc.) that will provide definitions of each role when clicked on.



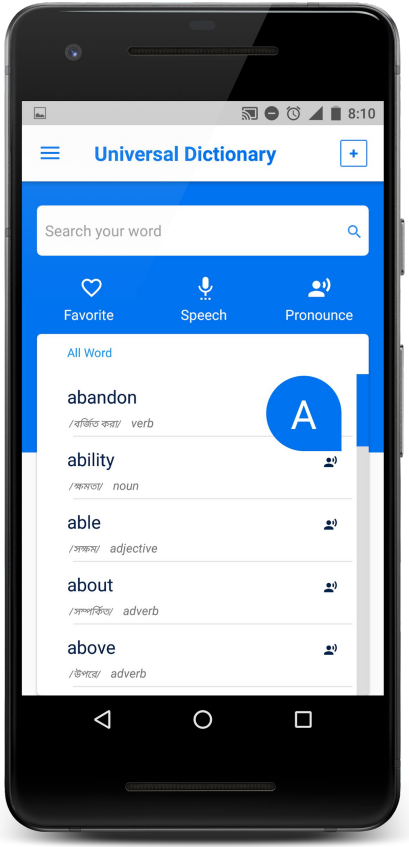
An example of a courtroom diagram.

Scrolling over the names of the courtroom members will show a definition / description

User App: Feature 2

- Glossary
  - A glossary page will contain hundreds of courtroom terminologies and definitions, and will be designed to facilitate learning and review.

Administrators will be able to add, modify, or remove terms and definitions



Admin Dashboard

- Login Page
  - Admins will be required to log in before modifying content
- Change Content System
  - Once logged in, the admin will be able to easily add, change, or delete vocabulary/definitions/text that appears in the iOS application
  - Enables admins to quickly update the content of the application without the need for a re-release through the Apple App store

Post-First Client Meeting

- After the client meeting, our team created a formal SRS document
  - Allows us to communicate our understanding of the system
  - Enables quick resolution of ambiguities before implementation begins
- We then discussed with the team from Discussion 1B who was also working with our client, and instead of competing with them, we decided to collaborate
  - Coordination amongst 11 people may be challenging
  - However, the collaboration will allow each team to focus more on their individual components
  - Ultimately results in a higher quality product being presented to the client
  - Reduces redundancy of development efforts
  - Received approval from Professor Eggert for collaboration

SRS Sample

3.1.2 User Class 2 - Team Administrators (i.e. Alicia's and Jyoti's administrative team)

3.1.2.1 Functional Requirement 2.1

ID: FR8

TITLE: Web application accessibility

DESC: The administrator should be able to navigate to the web application through a URL in their browser of choice.

RAT: To access the web application.

DEP: None

3.1.2.2 Functional Requirement 2.2

ID: FR9

TITLE: Administrator account registration

DESC: Registration will be done via invitation from an admin (presumably Alicia and Jyoti are the only admins for the time being).

RAT: To register an account on the web application

SRS Sample

3.1 Functional Requirements

3.1.1 User Class 1 - The Youths

3.1.1.1 Functional Requirement 1.1

ID: FR1

TITLE: Web application accessibility

DESC: The youth should be able to navigate to the web application through a URL in their browser of choice.

RAT: To access the web application.

DEP: None

3.1.1.2 Functional Requirement 1.2

ID: FR2

TITLE: Anonymous Access

DESC: The youth's access to the content should not be trackable, and should not be linked to their identity.

RAT: To protect the youth's lack of judicial knowledge from being used against them in court.

DEP: None

1A/1B Division of Labor

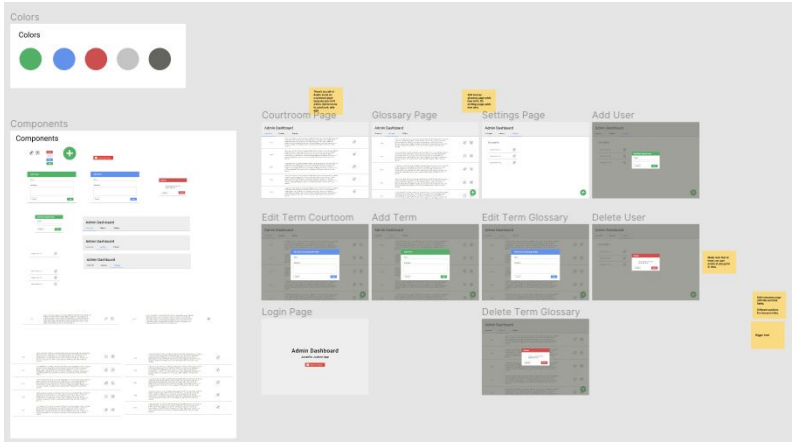
- Our team will work on:
  - ***Datastore***
    - database to store legal vocabulary
  - ***Admin application***
    - the website where the admins log in to update the data used by the user application
- Partner team will work on:
  - ***user application***
    - the iOS actual application with the court roles/roadmap/glossary
    - Ideally will be published to the App store
- Both teams will work together to agree on the API through which the user application will pull data from the backend

Second Client Meeting

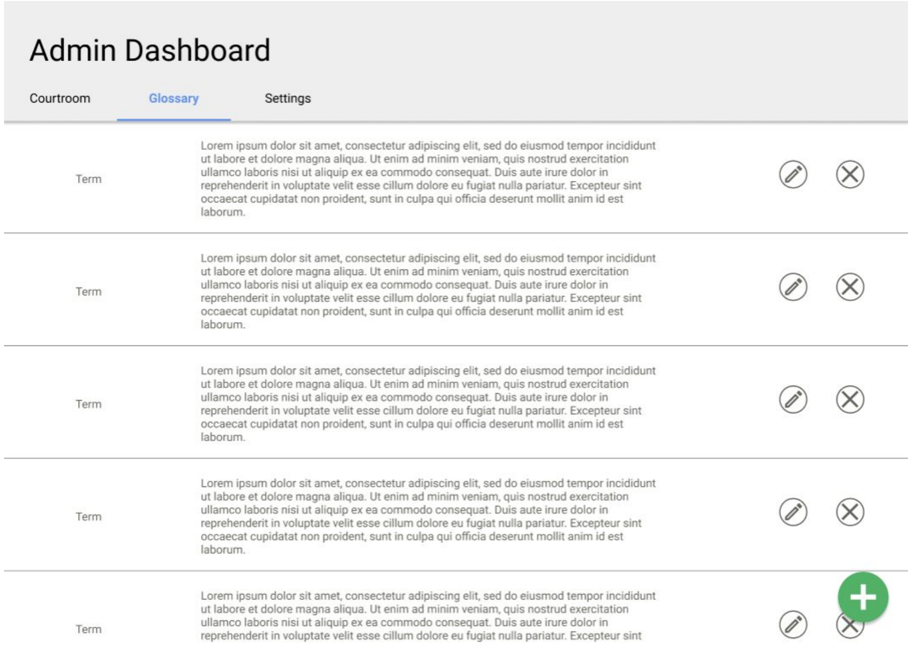
- On **May 15**, we met with the client again, this time with the Discussion 1B's team, where we presented our designs to the client.
- Results from the meeting:
  - Justice team is happy with the inviting UI of the user-facing iOS app, and the simplistic and functional UI of the admin dashboard
  - Reviewed SRS with justice team
  - Design and SRS have been agreed upon; implementation can begin

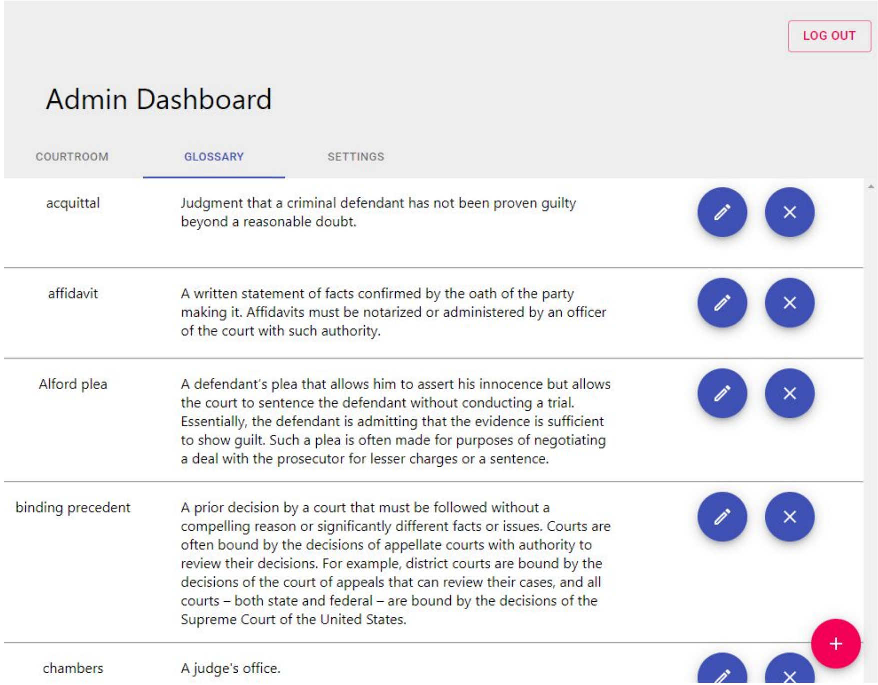
Design that was presented to clients

- We presented our design of the admin dashboard to the clients
  - See example pages on following slides
  - Source: https://www.figma.com/proto/dTylqQ3FWlXUTxDMuEOLtvpq/JJS-Project-Admin-Dashboard-Mockup?node-id=7%3A32&scaling=contain&redirected=1

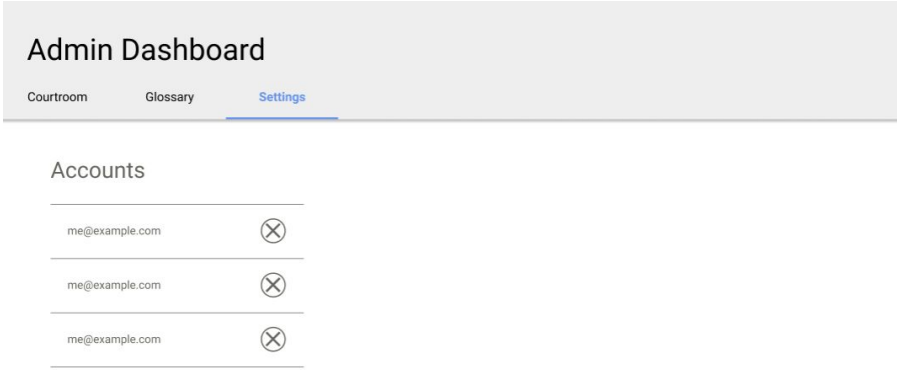


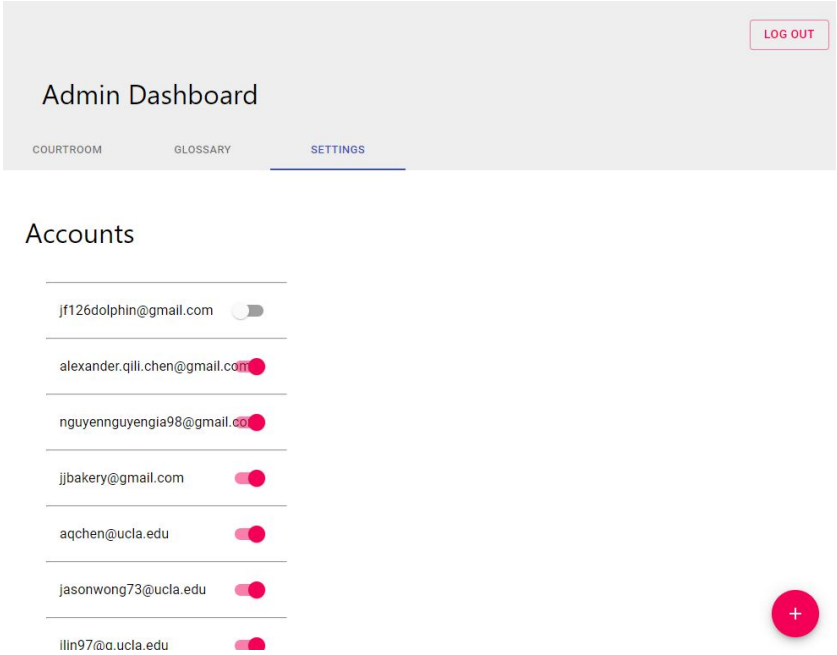
Designs vs Real Implementation



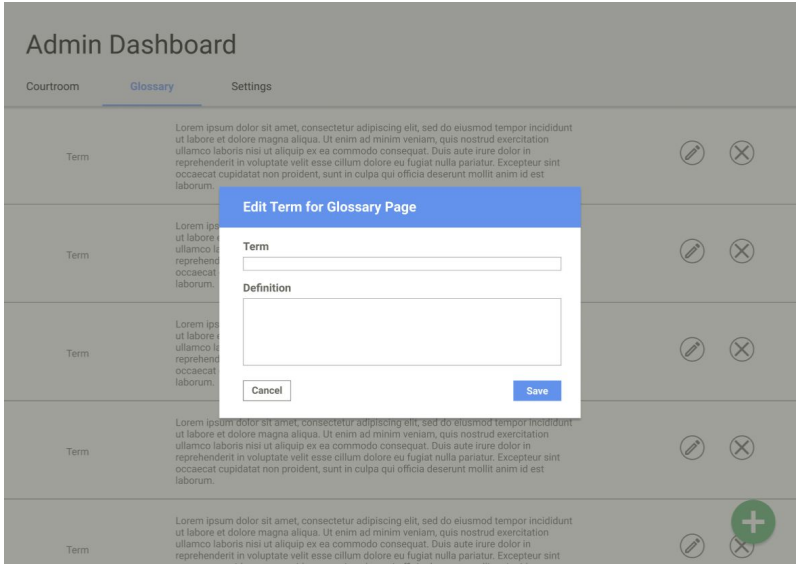


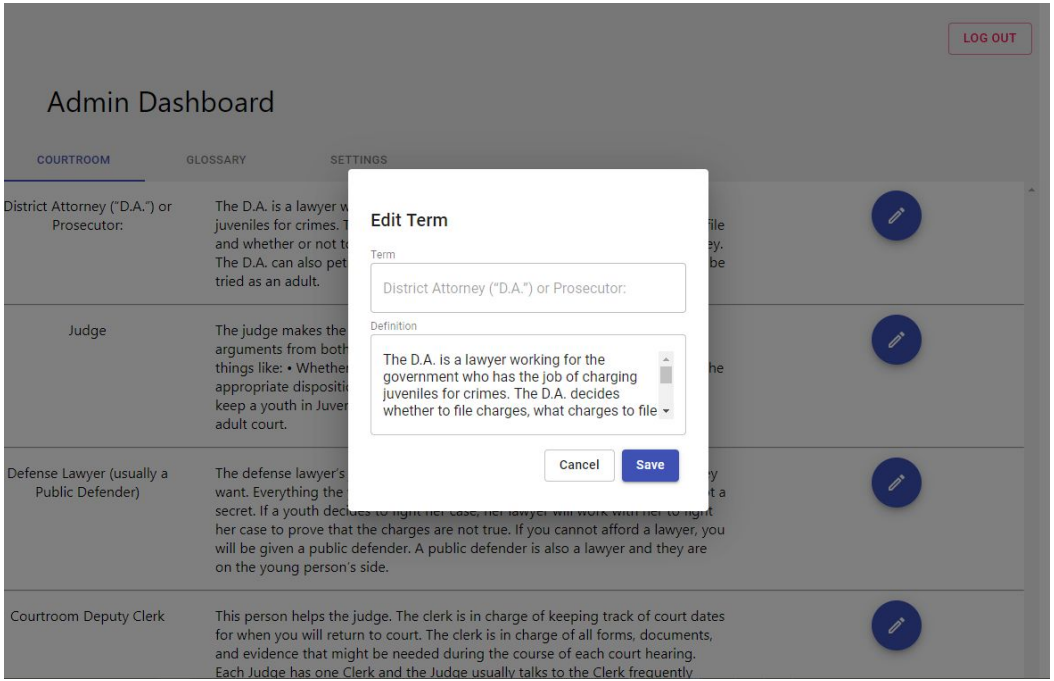
Designs vs Real Implementation





Designs vs Real Implementation



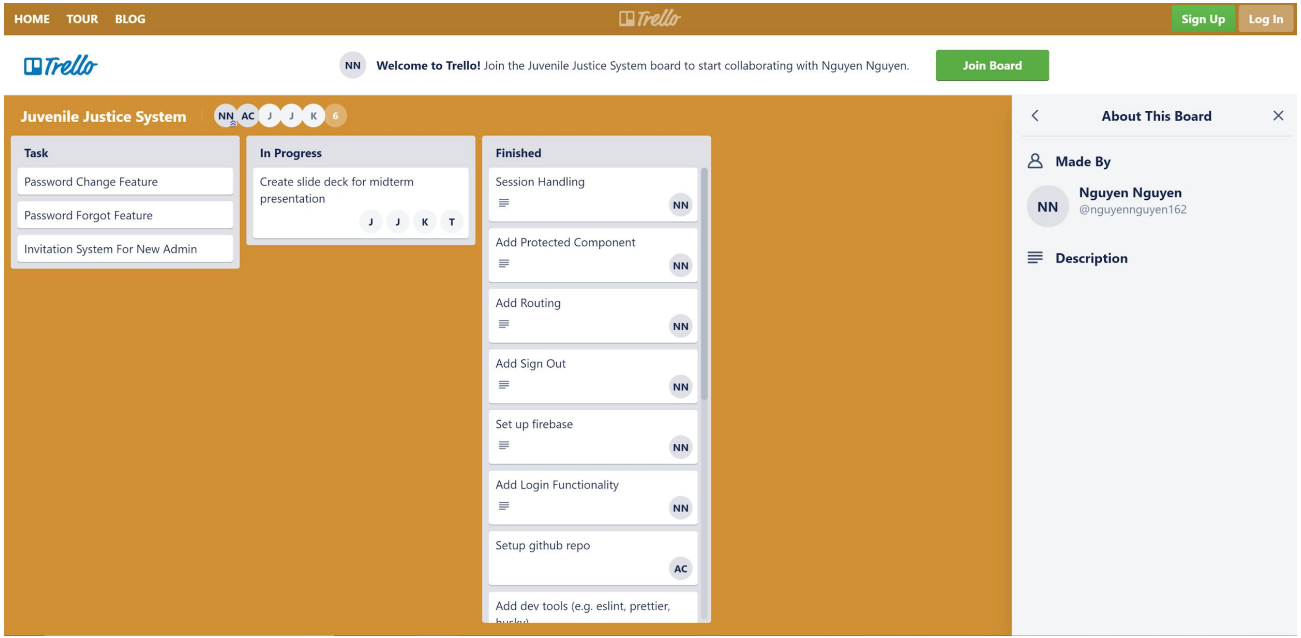


Post-Second Client Meeting

- After meeting with the client a second time, we set a date at the end of the month where we will present the finished product to them.

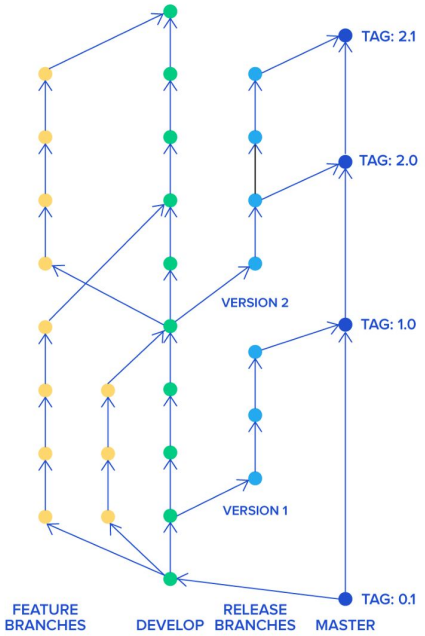
Software Engineering Processes: Task planning

- Task planning and allocation
  - We are using Trello to organize tasks amongst team members, and plan our development efforts



Software Engineering Processes: Version Control

- Version control
  - We are using GitHub to store our source code
  - Optimistic Version control (allows concurrent editing of modules)

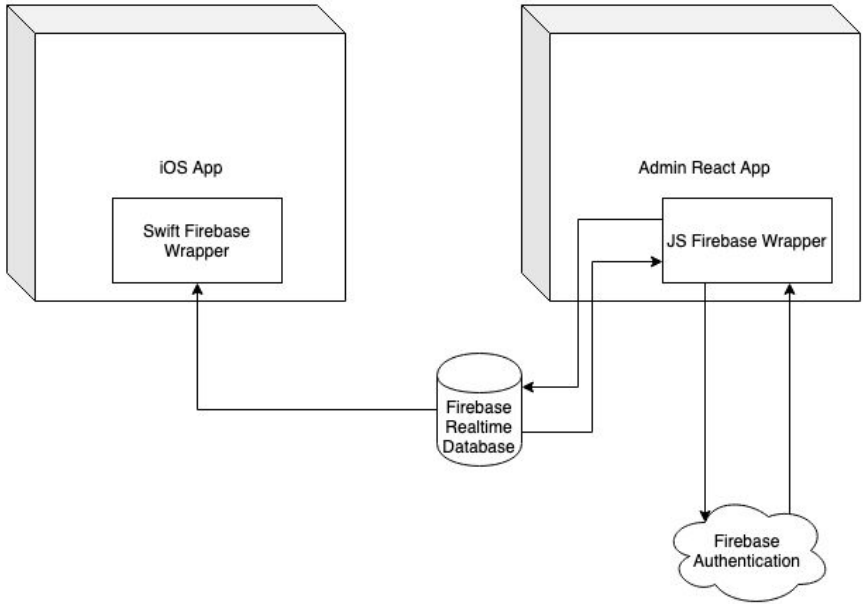


Software Engineering Processes: Code Style

- Eslint
  - Linter to ensure code style quality
- Prettier
  - Plugin that formats code consistently
- Husky
  - Sets up git hooks for eslint

Architecture of System

- After authenticating, justice system admins perform CRUD operations through the admin dashboard
  - Sends requests to the firebase realtime database
- Swift firebase wrapper exposes simple API to the iOS app (format TBD, but likely Get requests)
- iOS app reads vocabulary data from swift firebase wrapper, and presents it through the iOS app UI



Architecture/Deployment Decisions

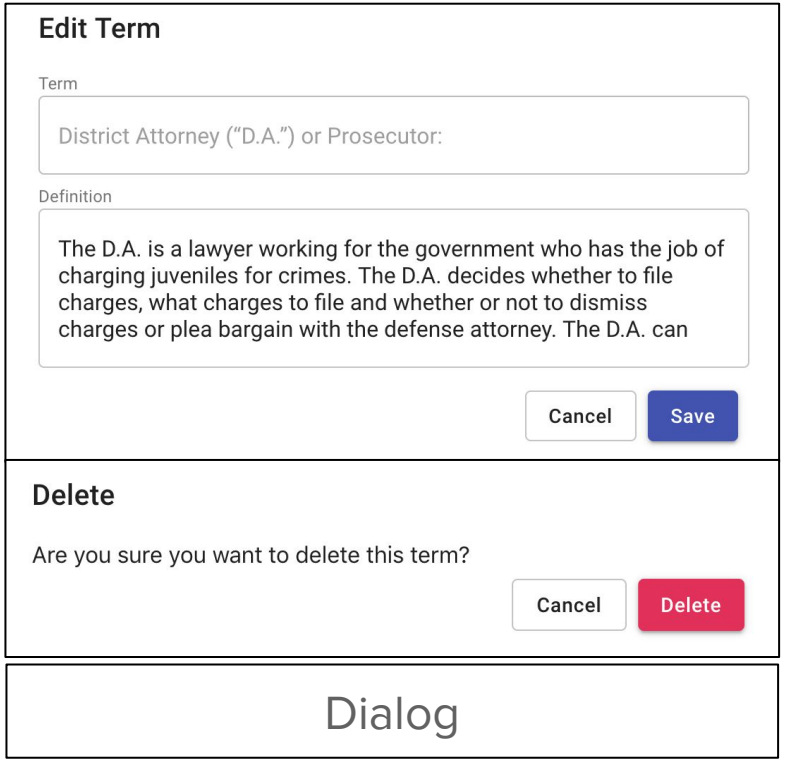
- Data Store: Firebase
  - Experienced with Firebase
  - Free quota, allows the justice system to continue serving their app with minimal/no operational costs
  - Amount of data stored is not projected to grow substantially as the app evolves
- Firebase hosting for serving the admin dashboard
  - Free, simple to use
  - Amount of admins is not projected to increase drastically
- Frontend: React
  - Experienced with React
  - React state will be relatively simple to manage
- Frontend Tests
  - Jest
  - Enzyme

Live Demo

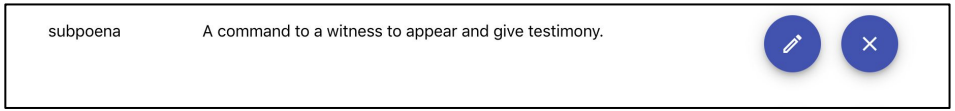
Frontend Development Takeaways

- Robustness of UI for various screen sizes ("responsive UI")
- Accessibility considerations (`aria`, color scheme to accommodate color blind users, etc)
- Modularity is key (code reuse, "buy vs build" components)
- Intuitive functionality (essence of Material UI)
- Design approval before frontend development is most effective
- Keep extensibility in mind (create method stubs and/or TODO comments for functionality that will be incorporated but currently not necessary)
- All other software engineering practices still in effect

Modularity & Code Reuse: Common Components



Dialog



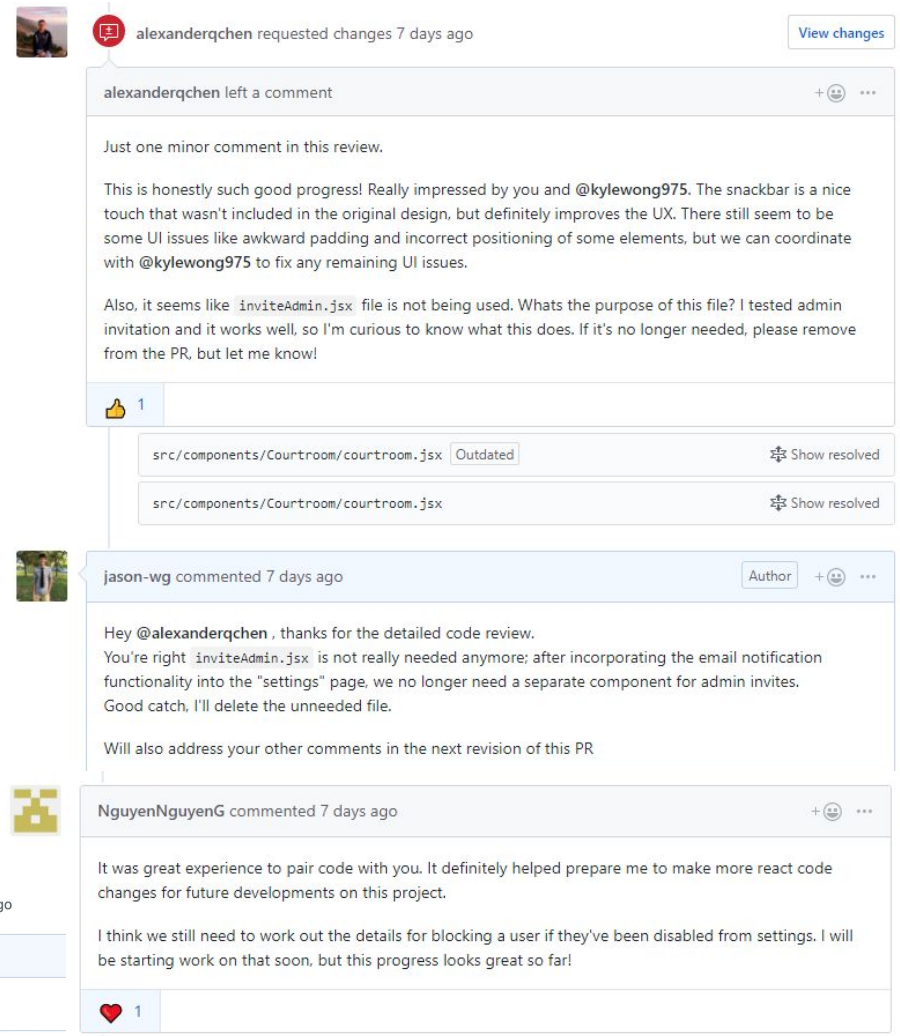
Term Card

Documentation

- Software requirements specification (constantly updated upon changing client needs)
- `README.md` for getting started and other relevant documentation
- Necessary to more efficiently onboard developers and users (for any less intuitive functionality, if any)

Software Engineering Processes: Collaborative Development and Peer Review

- Pair programming
- Tool-assisted peer reviews
  - Github code reviews
  - More formal
- Goal: Take advantage of strengths of both strategies
  - Tighter feedback loop of pair programming
  - More formal risk analysis and management than pair programming
    - All members are considered reviewers



Design Patterns and Code Organization

- Observer Pattern for syncing Backend Data Changes with Frontend
  - Easy to implement using Firebase
  - Subject:
    - Firebase Backend
  - Observers:
    - React components. On creation, they register with observer
- State Pattern
  - Object holds internal state
    - Fits nicely with ReactJS components, which use `state` and `props`
  - Object acts as a finite state machine
- Modularization via Components
  - Tied in very nicely with ReactJS's component structure
  - Made testing easier

Design Patterns and Code Organization

- Higher Order Component Design Pattern (HOC)

withAuthorization:

```
  componentDidMount() {
    const { firebase, history } = this.props;
    this.listener = firebase.auth.onAuthStateChanged(authUser => {
      this.setState({ authUser, display: true });
      if (!authUser && !isPublic) {
        history.push(ROUTES.LANDING);
      } else if (authUser && isPublic) {
        history.push(ROUTES.ADMIN_PANEL);
      }
    });
  }

  componentWillUnmount() {
    this.listener();
  }

  render() {
    const { authUser, display } = this.state;
    if (display) {
      if ((authUser && !isPublic) || (!authUser && isPublic)) {
        return <Component {...this.props} />;
      }
      return null;
    }
    return null;
  }
 }

 return withRouter(withFirebase(WithAuthorization));
};

export default withAuthorization;
```

Admin:

```
export default withStyles(styles)(withAuthorization(Admin)(false));
```

Landing:

```
const LandingPage = withAuthorization(Landing)(true);

function LandingExport() {
  const classes = useStyles();
  return <LandingPage classes={classes} />;
}

export default LandingExport;
```

Testing Frameworks

- Jest: JavaScript testing framework
- Enzyme: JavaScript Testing utility for React
  - Meant to be intuitive and flexible by mimicking jQuery's API for DOM manipulation and traversal

- Unit tests
- Snapshot tests
  - Whenever you want to make sure your UI does not change unexpectedly.
  - Renders a UI component, takes a snapshot, then compares it to a reference snapshot file stored alongside the test.
  - Test fails if the two snapshots do not match

Past Timeline

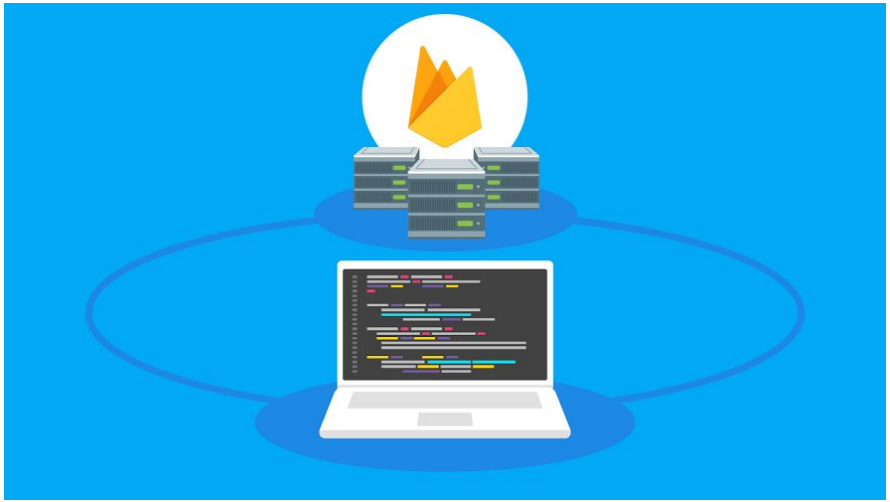
- 04/24: Project team was assigned
- 05/02: [Client meeting] First meeting with Alicia and Jyoti to discuss requirements
- 05/06: Finished first iteration of SRS
- 05/10: Decision to collaborate with associated team from Discussion 1B team for project
- 05/10: Internal team meeting to discuss design
- 05/13: Internal team meeting to review software design progress
- 05/15: [Client meeting] Second meeting with Alicia and Jyoti: review SRS and mock-ups
- 05/17: Midterm presentation
- 05/22: Finished back-end
- 05/27: [Client meeting] Deliver MVP to Alicia and Jyoti
- 05/30: Finalize stretch goals and freeze code
- 05/31: [Client meeting] Project demo
- 06/05: Final class presentation
- 06/10: Deadline to submit final report

Future Plans

- i18n (Internationalization)
  - Have a language setting, to display content in Spanish for example
  - Especially applicable in the LA area, with native Spanish speakers
- Showing preview of changes
  - Before changes are made viewable to the youth, the administrators can see what the changes will look like on the mobile phone (possibly through an emulator)
- Linking terms to definition in glossary
  - Terms that are included as part of definitions for other terms in the courtroom should be linked to their definition in the glossary

Reflection

- Having a serverless architecture is nice ...
  - Able to do all CRUD operations with Firebase database in front-end with ReactJs
  - Did not have to spin up servers and control traffic with load balancers
  - Did not have to worry about scaling issues



Reflection

- But it has its limitations ...

```
const { email, password, users, usersObject } = this.state;
const { firebase, history } = this.props;

// Key into the data from firebase
const emailKey = email.replace(/\./g, ',');

if (!users) {
  this.setState({
    error: new Error("Unable to fetch firebase user list")
  });
  return;
}

// If user is not in auth DB
if (users.indexOf(emailKey) === -1) {
  this.setState({
    error: new Error(
      "Not authorized. This email does not have an admin account."
    )
  });
  return;
}

// User is in the auth DB, but they have been disabled
if (!usersObject[emailKey].enabled) {
  this.setState({
    error: new Error(
      "Not authorized. The admin account associated with this email has been disabled. Contact your manager to
    )
  });
  return;
}
```

```
// If user already exists, do not allow the add
if (users.indexOf(emailInput.replace(/\./g, ',')) !== -1) {
  this.handleShowSnackbar("Admin with this email already exists");
  return;
}

firebase
  .doCreateNewUser(emailInput)
  .then(() => firebase.doPasswordReset(emailInput))
  .then(() => {
    this.writeUserToDb(emailInput, true);
  })
  .then(() => {
    this.setState({
```

Delete a user

The Firebase Admin SDK allows deleting existing users by their `uid`:

Node.js | Java | Python | Go

```
admin.auth().deleteUser(uid)
  .then(function() {
    console.log('Successfully deleted user');
  })
  .catch(function(error) {
    console.log('Error deleting user:', error);
  });
```

Thank you

Questions / comments