Simulence

Team 34

Gilbert Ye, Ali Gencoglu, Taehwan Park, Ramzi Dajani, Ahmad Islah, Caules Ge, Sonny Chen

Temporary Slide to Assign Roles

Problems & Users & Partner : Gilbert

Solutions & Learnings: Ali

Demo : Ted

Team Process & Access & Hand-Off : Ramzi

Tech Stack & Architecture : Ahmad

Design : Caules

Team Workflow(key learning) : Sonny

What is Simulence?

- An application where game developers can affordably playtest their own games without having to enlist an outside company.
  - Users: Developers, Testers, Admin/Partner
- Problems that Simulence addresses:
  - For game developers:
    - Provide a cheap and reliable system to playtest their games themselves.
    - Store meaningful data on their games.
  - For Testers:
    - Provide a simple system for matching willing testers to developers and games to test.
- Our team specializes on the tester matching aspect of the application.

Problems & Users & Partner

- Continuation of development of previous work. Issues from previous team:
  - Deployment
  - Security of login feature
  - Slow loading time
  - Bugs
- Areas for improvement of the application:
  - UI/UX
  - Developers cannot quickly search for testers
  - Testers have no way of expressing their general interests and habits

Solutions

Porting the previous project from Axios/MongoDB to Firebase

- Secure login
- Faster loading times
- Less cost for the partner
- Less overhead for developers

Custom firestore access interface

- More scalable and reliable
- Documentation

Global variables for UI design (WIP)

Many bug fixes

Demo

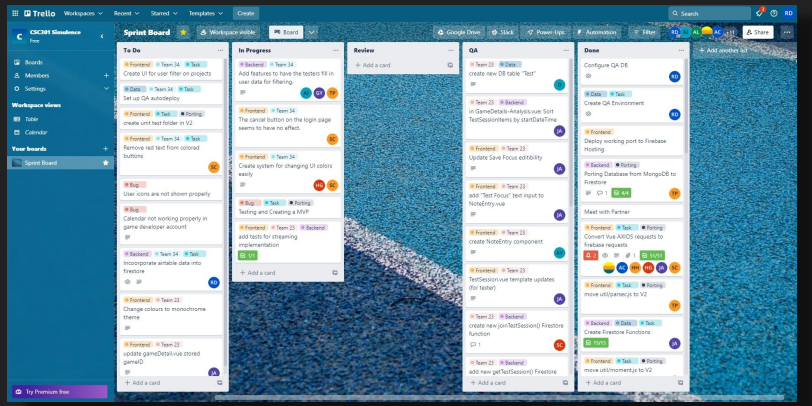
1. On the Scene
    a. Faster loadings
    b. Advanced searching for play testers
    c. UI/UX improvements in progress
    d. QA Environment

2. Behind the Scenes
    a. Secure authentication
    b. Numerous handled edge cases

Team Process

- Tasks assigned on Trello using scrum
- Scrum Master/Product Manager assign priority to tasks
- Other team members code review on github and test changes in QA
- All PRs deployed to QA branch auto-deploy to the QA environment
- Once testing is successful, task is marked as “done”

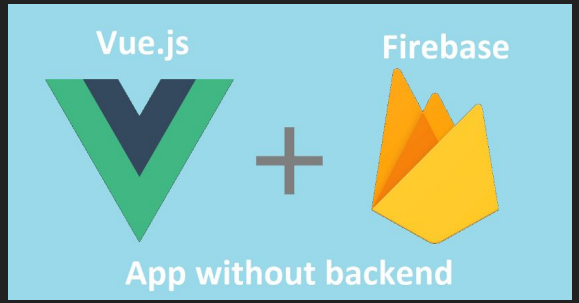
```yaml
name: Deploy to QA Channel

on:
  push:
    branches:
      - qa-34

jobs:
  deploy_live_website:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      # Add any build steps here. For example:
      - run: npm run build
      - uses: FirebaseExtended/action-hosting-deploy@v0
        with:
          repoToken: "${{ secrets.GITHUB_TOKEN }}"
          firebaseServiceAccount: "${{ secrets.FIREBASE_SERVICE_ACCOUNT }}"
          projectId: simulence-qa
          channelId: live
```



Access & Hand-Off

- Our partner can access our project via live URL
  https://simulence-e7915.web.app
- Partner also has access to firebase for future deployment.
- Once the project is complete, full github permissions would be granted to the partner
- Easy 2-step deploy:
  npm run build & firebase deploy

Tech Stack



- Frontend: Vue, HTML/CSS, JavaScript
- Backend/Database: Cloud Firestore
  - NoSQL Database
- Interacting with the database:

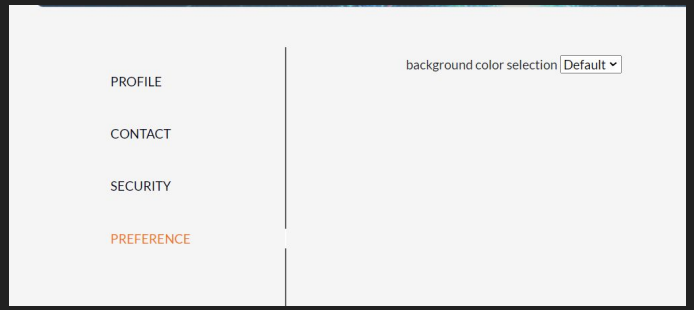
Cloud Firestore → Firestore.js file → *.vue Component files

- Firebase also handles authentication and hosting

Design

Based on the design from previous teams in CSC301.

New features for design:



Front end:

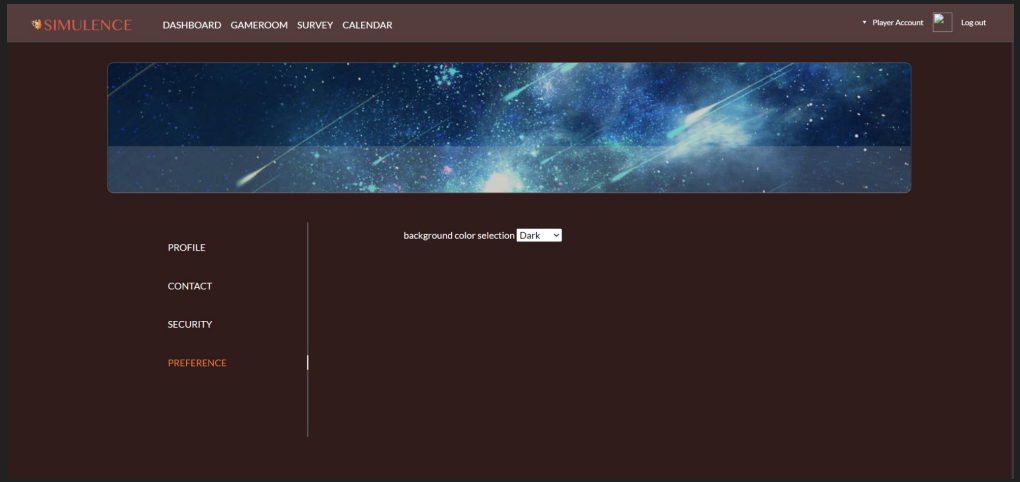
- Created global variables to store UI colors. After users selected a color theme in the userProfile component, the corresponding color variable will be passed to components (eg. NavigationBar) to reset their color.
- Added the user preference page in the userProfile component to enable users to select their preferred color theme for the UI.

Back end:

- Have a separate file (Firestore.js) which holds all JS methods that interact with Firestore DB, and these methods are then called from the component Vue files

global variables to store UI colors

```
export default new Vuex.Store({
    state: {
        // Login page
        logincolor: '#E39E4D',
        cancelbcolor: '#fbfbfb',
        textcolor: '#3b3a3a',
        promptcolor: '#4C4F55',
        promptbkcolor:'#FFFFFF',
        loginpage:'#F5F5F5',
        logincontainer:'#7C7E82',
        logintitle:'#FFFFFF',

        // Loading page
        loading:'#FC7727',

        // Navigation bar
        navtheme:'#F8DC77',
        navctn:'#4C4F55', // underline navigation bar
        naveffect:'#FC7727', // button effect
        triangle:'#FC7727',
        account:'#FC7727', // account name
        logout:'#FC7727',
```



Architecture

Code snippet for creating a Tester

(corresponding function in Firestore.js and

being called in AddTesterItem.vue)

```
import firebase from 'firebase'
import {firestore, query, collection, where, getAll} from 'firebase/firestore'
import {db, auth} from './main';

// AddTesterItem.vue
async function createTester(gameId, testerId) {
    function err(e) {
        return {success: false, error: e};
    }

    // check if user exists
    return new Promise((resolve, reject) => {
        // check if game exists
        db.collection("Game").doc(gameId).get()
        .then(doc => {
            if (!doc.exists) {
                // return {success: false, message: "game does not exist"};
                reject({success: false, error: "game does not exist"});
            }

            const gameTesterIDs = doc.data().testerIDs

            // if the user exists and is not in the game, add the user to it
            db.collection("User").doc(testerId).get()
            .then(doc => {
                // console.log(doc.exists);
                if (!doc.exists) {
                    console.log("user doesn't exist");
                    // return {success: false, message: "user does not exist"};
                    reject({success: false, error: "user does not exist"});
                }

                if (gameTesterIDs.includes(doc.id) || doc.data().userType !== "tester") {
                    reject({success: false, error: "user is not a tester or is already in the game"});
                } else {
                    db.collection("Game").doc(gameId).update({

                        testerIDs: [...gameTesterIDs, testerId]
                    })
                    .then(res => {
                        // return {success: true};
                        resolve({success: true})
                    }).catch(e => reject(err(e)));
                }
            }).catch(e => reject(err(e)));
        }).catch(e => reject(err(e)));
    });
}
```

```
methods: {
    addTester: function() {
        let that = this;
        that.isLoading = true;
        Firestore.createTester(this.gameId, this.id)
        .then(res => {
            console.log("add tester");
            that.$emit('addTester');
            that.isLoading=false;
        })
        .catch(e => {
            alert(e.error);
            that.isLoading=false;
        });
    }
}
```

Key Learnings

Our workflow:

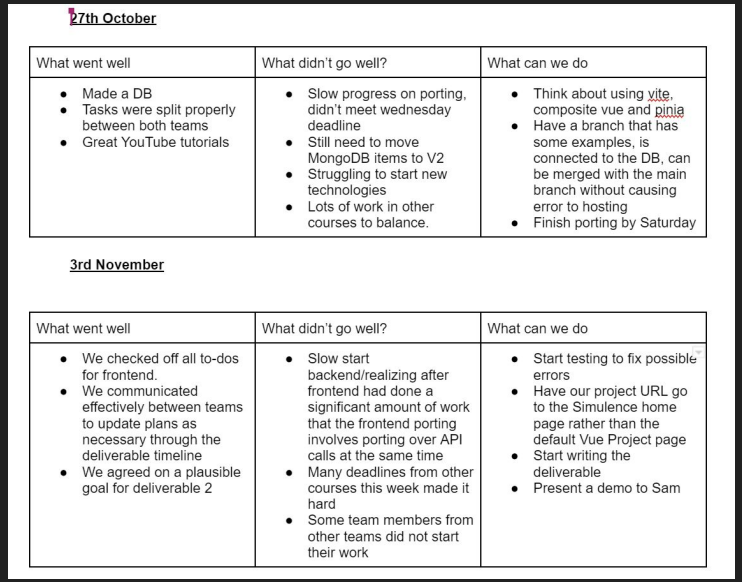
- Weekly meetings with our partner.
- SCRUM meetings within our team.
- Teams collaborate and coordinate using Trello Board

Midway changes:

- Flexible meeting times
- Flexible roles

Takeaways:

- Build on top vs. build from scratch.
- Teamwork and communication.



27th October

What went well	What didn't go well?	What can we do
• Made a DB • Tasks were split properly between both teams • Great YouTube tutorials	• Slow progress on porting, didn't meet wednesday deadline • Still need to move MongoDB items to V2 • Struggling to start new technologies • Lots of work in other courses to balance.	• Think about using vite, composite vue and pinia • Have a branch that has some examples, is connected to the DB, can be merged with the main branch without causing error to hosting • Finish porting by Saturday

3rd November

What went well	What didn't go well?	What can we do
• We checked off all to-dos for frontend. • We communicated effectively between teams to update plans as necessary through the deliverable timeline • We agreed on a plausible goal for deliverable 2	• Slow start backend/realizing after frontend had done a significant amount of work that the frontend porting involves porting over API calls at the same time • Many deadlines from other courses this week made it hard • Some team members from other teams did not start their work	• Start testing to fix possible errors • Have our project URL go to the Simulence home page rather than the default Vue Project page • Start writing the deliverable • Present a demo to Sam

Thank you for listening to our presentation

Any questions?

Individual Contribution

Taehwan Park: Handled diverse edge cases that were prevalent when browsing/filtering/adding/removing testers of game project. Fixed errors that emerged during porting (mostly relevant to the CRUD)

Ali Gencoglu: Worked on building the server query architecture for the frontend, porting the HTTP calls from previous team's solution to firebase. Added documentation, handled bug fixes and issued code reviews.

Ahmad Islah: Worked on implementing the searching and filtering of adding testers to a project in the backend

Sonny Chen: Worked on implementing changing color themes for UI using vuex.

Caules Ge: Implemented the user Preference page to enable users to select their color preference in the setting.

Gilbert Ye: Implemented the PlayTesterAttribute page where users have a variety of attributes that they can assign themselves to. These attributes are stored in the database and then used to filter the testers later.

Ramzi Dajani: Led team meetings as scrum master and implemented a QA environment to increase testing productivity