Bookee

Session: 2021 – 2025

Submitted by:

Danish Akram Gondal 2021-CS-123

M Jawad Haider 2021-CS-149

Supervised by:

Ma’am Maida

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

CONTENTS

INTRODUCTION:

Bookee is a web-based application. It provides a convenient way for users to read different books in this tech era. It is a place where you can see what your friends are reading and vice versa. You can create bookshelves to organize what you've read (or want to read). You can comment on each other's reviews. You can find your next favorite book. And on this journey with your friends you can explore new territory, gather information, and expand your mind.

PROJECT FEATURES:

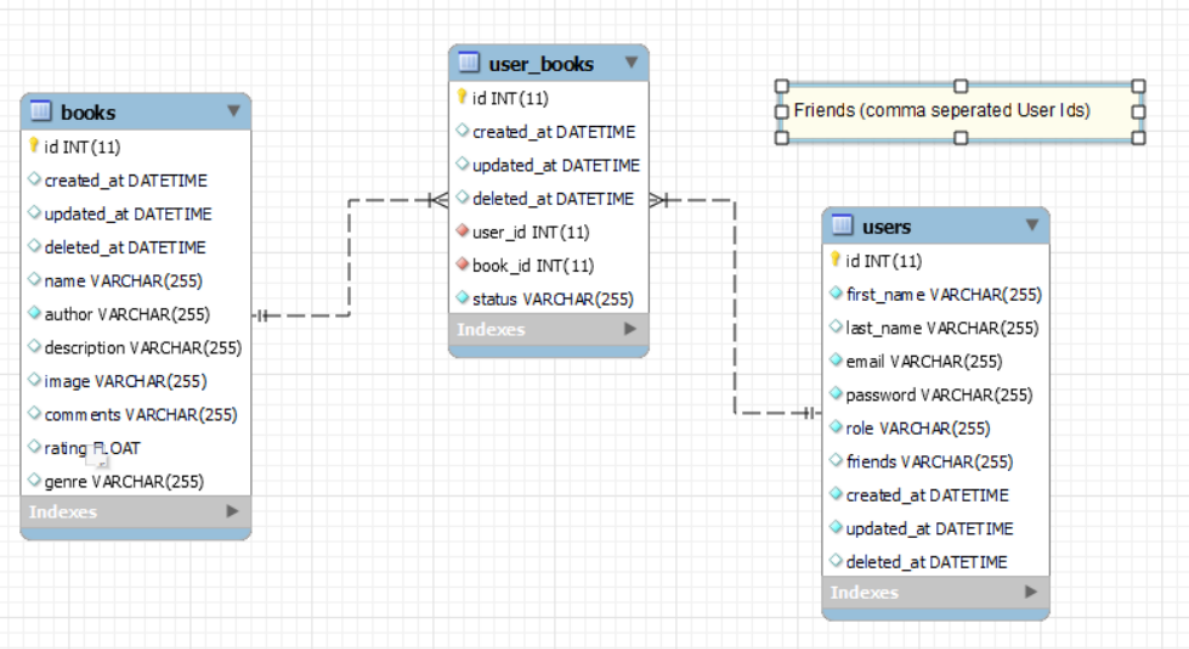
New user can sign up if new to the application. Further he can add up to multiple books in order to keep record of his books and readings. Currently the application consists of only 16 to 20 books but this can be expanded. The user can view the Community of the application users can view few actions of the users, he can follow the people and can also add the book suggested by them. Further the user can follow any user of this application. Currently the scope of the project is limited but further features can be implemented like the user can see his own virtual Book Shelf and keep records of all of his books.

USE CASES:

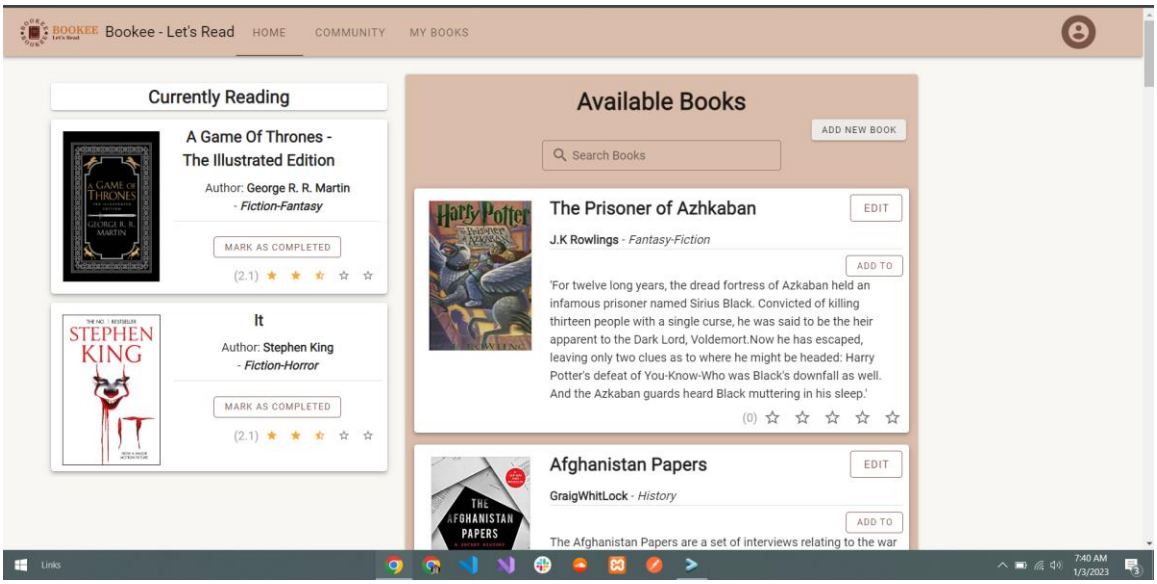
The Application can be used by any age group who has habit of reading books. This application allows the users to efficiently keep the records of his books. It is a public oriented application which is open to every user and anyone can easily access it. Moreover the application can also be introduced in the Universities where most of the students are unable to find the relevant course books. For this this application can be enhanced by adding categories for different types of books and provide a path to find that book.

CLASS DIAGRAM:

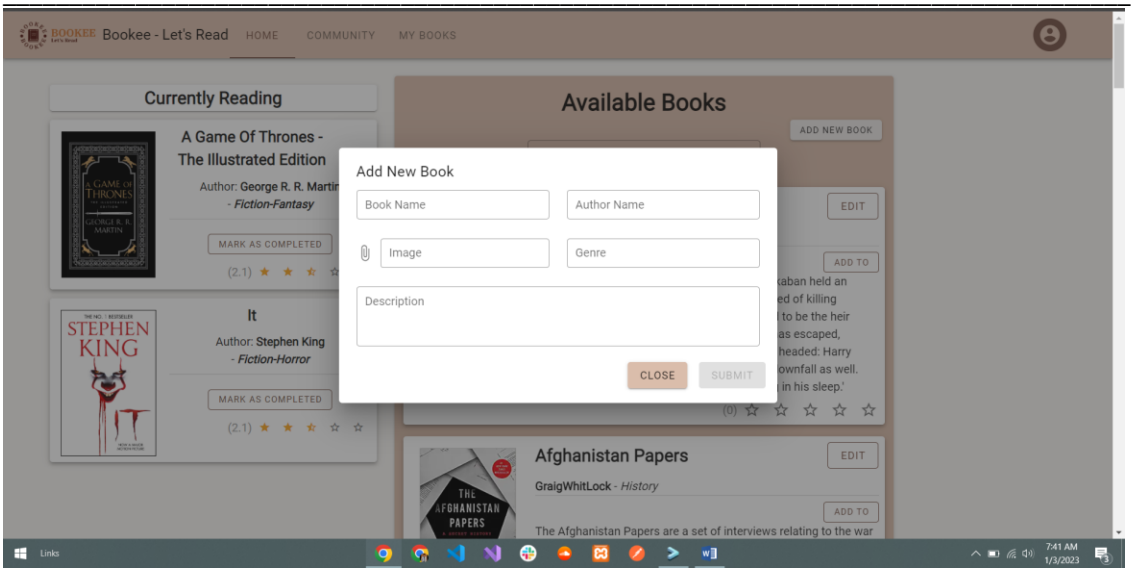
A junction table is used between the User and Books. And the following relation is maintained as comma-separated friendIds in User table.

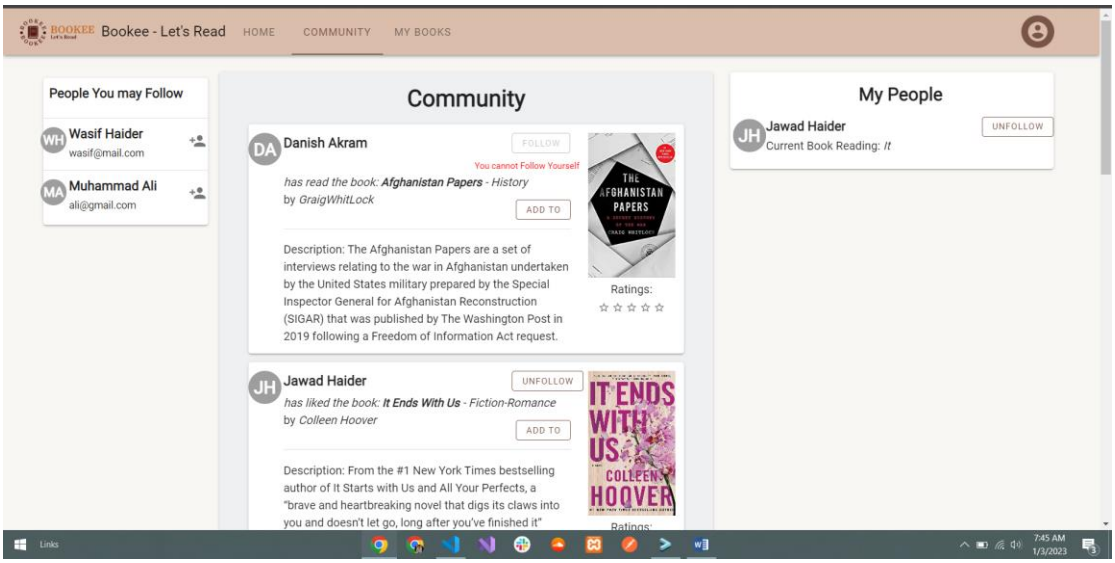


USER INTERFACE DESIGN:

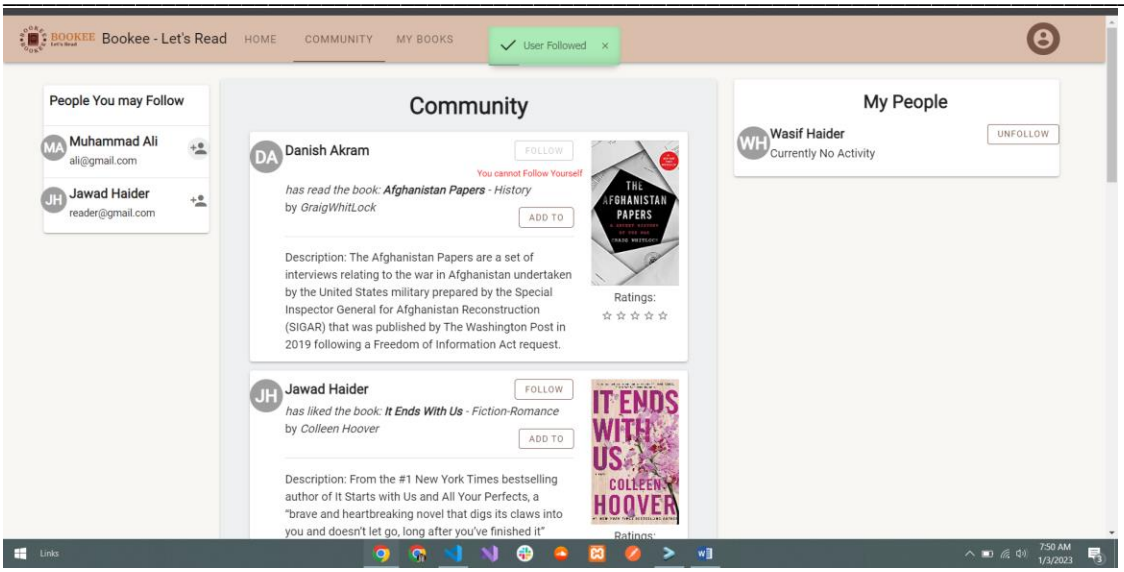


Danish Akram Gondal 2021-CS-123
M Jawad Haider 2021-CS-149





Danish Akram Gondal 2021-CS-123
M Jawad Haider 2021-CS-149



TECHNOLOGY AND RESOURCES:

Fronted is built using Vue framework of javascript using multiple packages including: Vuetify

Vue-Router, Vuex, Izitoast, and further Axios is used as a pathway between frontend and backend connection.

Backend is built using Yo and generator-varnode, the package creates brief backend with Node.js,

MySql Database is used with sequelize and express.js and jsonwebtoken is used for hashing passwords in database.

Vuetify: https://vuetifyjs.com/en/

Vue-router: https://router.vuejs.org/

Axios: https://axios-http.com/

Generator-varnode: https://github.com/isajjadali/generator-varnode

Sequelize: https://sequelize.org/

CODE:

FRONTEND

GRAPH.JS (CLASS)

```
class Graph {
        constructor() {
                this.AdjList = new Map();
```

```
        this.admin = {
            firstName: 'Danish',
            lastName: 'Akram',
            email: 'admin@gmail.com',
            role: 'admin',
            password: 'sajjad734',
        };
        const reader = {
            firstName: 'Jawad',
            lastName: 'Haider',
            email: 'reader@gmail.com',
            role: 'reader',
            password: 'sajjad734',
        };
        this.addNewUser(this.admin);
        this.addNewUser(reader);
    }
    addNewUser(newUser) {
        this.AdjList.set(newUser.email, {
            user: newUser,
            friends: [],
        });
    }
    addFriends(currentUser, friend) {
        const { user, friends } = this.AdjList.get(currentUser.email);
        if (friends.find((f) => f.id === friend.id)) {
            return false;
        }
        friends.push(friend);
        return true;
    }
```

```
removeFriend(currentUser, friend) {
        let { user, friends } = this.AdjList.get(currentUser.email);
        friends = friends.filter((f) => f.id !== friend.userId);
        this.AdjList.set(user.email, {
                user,
                friends,
        });
}
findUserByEmail(email) {
        const isUserFound = this.AdjList.get(email);
        return isUserFound.user;
}

userLogin(user) {
        const isUserFound = this.AdjList.get(user.email);
        if (isUserFound && isUserFound.user.password === user.password) {
                return isUserFound.user;
        }
        return false;
}
isUser(user) {
        return this.AdjList.has(user.email);
}
printGraph() {
        this.AdjList.forEach((obj) => {
                const user = obj.user;
                let str = '';
                str += user.firstName + ' -> ';
                const friends = obj.friends;
                friends.forEach((neighbour) => {
                        str = str + neighbour.firstName + ' ';
```

```
                });
                console.log(str);
            });
        }
}
export default new Graph();
```

MAIN.JS

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import VueIziToast from 'vue-izitoast';
import 'izitoast/dist/css/iziToast.min.css';
import axios from 'axios';

import App from './App.vue';
import vuetify from './plugins/vuetify';
import routes from './routes';
import store from './store';

Vue.config.productionTip = false;

axios.defaults.baseURL = process.env.VUE_APP_SERVER_URL;

// Vue Router Documentation https://v3.router.vuejs.org/installation.html#npm
const router = new VueRouter({
    mode: 'history',
    routes,
});

router.beforeEach((to, from, next) => {
    const token = localStorage.getItem('token');
    let user = JSON.parse(localStorage.getItem('user') || '{}');
```

```
        if (to.path === '/login') {
                return next();
        }
        if (
                to.matched.every((rec) => {
                        const roles = rec.meta.roles || [];
                        if (rec.meta?.authNotRequired) {
                                return next();
                        }
                        return !roles.includes('all') && !roles.includes(user.role);
                })
        ) {
                return next({ path: '/home' });
        }
        next();
});

Vue.config.errorHandler = (err, vm) => {
        // err: error trace
        // vm: component in which error occured
        // info: Vue specific error information such as lifecycle hooks, events etc.

        console.error(err, '<=== Main.js Error Tracking');
        if (err) vm.$toast.error(err.data.error?.body || err.data.error);
};

const options = {
        position: 'topCenter',
};
Vue.use(VueIziToast, options);
```

```
Vue.use(VueRouter);
// Vue.mixin(pageTitleMixin);

Promise.all([store.dispatch('fetchUsersData')]).then((res) => {
	new Vue({
		vuetify,
		router,
		store,
		render: (h) => h(App),
	}).$mount('#app');
});
```

GLOBAL.JS

```
import axios from 'axios';
import UsersGraph from './Graph';
/* eslint-disable */

const AuthToken = localStorage.getItem('token');
axios.defaults.headers.common['Authorization'] = AuthToken;
axios.interceptors.request.use(
	(config) => {
		const token = localStorage.getItem('token');
		if (token) {
			config.headers.common['Authorization'] = `Bearer ${token}`;
		}
		return config;
	},
	(error) => Promise.reject(error)
);

export const state = () => ({
```

```
        user: null,
        books: [],
        searchQuery: '',
        bookReading: [],
        userBooks: [],
        friends: [],
        community: [],
        allUsers: [],
});
export const actions = {
        async getMe({ commit }) {
                // const user = localStorage.getItem('user');
                // if (user) {
                //     commit('SET_ME', JSON.parse(user));
                // } else if (localStorage.getItem('token')) {
                const response = await axios.get('/me');
                commit('SET_ME', response.data.data.user);
                return response.data.data.user;
                // }
        },

        async login({ commit, state }, payload) {
                const user = UsersGraph.userLogin(payload);
                const response = await axios.post('/login', payload);
                commit('SET_USER', response.data.data);
        },

        async signUp({ commit }, payload) {
                const response = await axios.post('/signup', payload);
                commit('SET_NEW_USER_IN_GRAPH', response.data.data.user);
                return response;
```

```
    },

    async fetchUsersData({ commit, state }, onAppStart = true) {
        const response = await axios.get(`/all-users`);
        let users = response.data.dataItems;
        if (onAppStart) {
            users.forEach((user) => {
                commit('SET_NEW_USER_IN_GRAPH', user);
                if (user.friendsList?.length) {
                    user.friendsList.forEach((friend) => {
                        UsersGraph.addFriends(user, friend);
                    });
                }
            });
            UsersGraph.printGraph();
        } else {
            users = users
                .map((user) => {
                    if (
                        +user.id === +state.user.id ||
                        state.user.friends?.includes(user.id)
                    ) {
                        return null;
                    }
                    return user;
                })
                .filter((u) => !!u);
        }
        commit('SET_ALL_USERS', users);
        return users;
    },
```

```
async fetchLoggedInUser({ commit }, forceRefresh = false) {
        if (!forceRefresh && state.user) {
                return;
        }
        const response = await axios.get('/me');
        commit('SET_LOGGED_IN_USER', response.data);
        return response.data;
},
//================================== Books Actions
async fetchAllBooks({ commit, state }, forceRefresh = false) {
        if (!forceRefresh && state.books.length) {
                return;
        }
        const response = await axios.get('/all-books');
        commit('SET_BOOKS_LIST', response.data.dataItems);
        return response.data.dataItems;
},

//================================== Admin Calls
async updateBook({ commit }, book) {
        const response = await axios.put(`/admin/books/${book.id}`, book);
        commit('UPDATE_BOOK', response.data);
},

async getBook({}, id) {
        const response = await axios.get(`/admin/books/${id}`);
        return response.data;
},

async createBook({ commit, state }, book) {
```

```
            const response = await axios.post('/admin/books', book);
            return response;
        },

        //================================== Friends Calls
        async getAllFriends({ commit }, userId) {
            const response = await axios.get(`/all-friends/${userId}`);
            let allFriends = response.data.dataItems;
            if (response.data.count) {
                allFriends = allFriends.map((friend) => {
                    const user = friend.User ? friend.User : friend;
                    user.fullName = convertFullName(user.firstName, user.lastName);
                    return {
                        ...user,
                        book: { ...friend.book, status: friend.status } || {},
                        id: friend.id,
                        userId: friend.userId || friend.id,
                        bookId: friend.bookId,
                    };
                });
                commit('SET_ALL_FRIENDS', allFriends);
                return allFriends;
            }
            commit('SET_ALL_FRIENDS', []);
            return response.data;
        },

        async addNewFriend({ commit, state }, data) {
            const isAdded = UsersGraph.addFriends(data.user, data.friend);
            UsersGraph.printGraph();
            if (!isAdded) {
```

```
                        return isAdded;
                }
                state.friends.push(data.friend);
                const response = await axios.put(`/add-friend/${data.user.id}`, {
                        userId: data.user.id,
                        friendId: data.friend.id,
                });
                return isAdded;
                // return response.data.data;
        },

        async removeFriend({ commit, state }, data) {
                UsersGraph.removeFriend(data.user, data.friend);
                UsersGraph.printGraph();
                const response = await axios.put(`/remove-friend/${data.user.id}`, {
                        userId: data.user.id,
                        friendId: data.friend.userId,
                });
                // return response.data.data;
        },

        async fetchCommunity({ commit }) {
                const response = await axios.get('/community');
                commit('SET_COMMUNITY', response.data.dataItems);
                return response.data.dataItems;
        },

        //=================================== User Books Calls
        async addUserBook({}, data) {
                const response = await axios.post('/add-book', data);
                return response.data;
```

```
        },

        async getBookReading({ commit, state }, userId) {
            const response = await axios.get(`/book-reading/${userId}`);
            commit('SET_BOOK_READING', response.data.dataItems);
            return response.data.dataItems;
        },

        async getAllUserBooks({ commit }, userId) {
            const response = await axios.get(`/user-books/${userId}`);
            const bookReading = response.dataItems.find((book) => book.isReading);
            commit('SET_BOOK_READING', bookReading);
            commit('SET_ALL_USER_BOOKS', response.dataItems);
        },
};

function convertFullName(firstName, lastName) {
        firstName = firstName.charAt(0).toUpperCase() + firstName.slice(1);
        lastName = lastName.charAt(0).toUpperCase() + lastName.slice(1);
        return `${firstName} ${lastName}`;
}

export default {
        state,
        mutations,
        actions,
        // getters,
};
```

APP.VUE

```
<template>
        <v-app>
```

```
                <v-main class="default-background">
                        <router-view />
                </v-main>
        </v-app>
</template>

<script>
export default {
        name: 'App',
};
</script>

<style>
body,
.default-background {
        /* background-color: #eff0f2; */
        background-color: #f9f7f4;
}
</style>
```

LOGINPAGE.VUE

```
<template>
        <v-app>
                <v-container class="bg" fluid fill-height>
                        <v-row class="justify-center">
                                <v-col cols="12" md="12" sm="12">
                                        <v-img
                                                :src="require('../assets/logo.svg')"
                                                class="my-3"
                                                contain
                                                height="150"
                                        />
```

```
                    </v-col>


                    <v-row class="text-center">
                        <v-col cols="12" md="12" sm="12">
                            <h3 class="font-italic">
                                Welcome to Bookee. Let's read, as reading heals!
                            </h3>
                        </v-col>
                    </v-row>
                </v-row>
                <v-row class="mt-8 justify-center">
                    <v-card
                        :width="$vuetify.breakpoint.mobile ? '80%' : '35%'"
                        color="#EFEBE9"
                        elevation="7"
                    >
                        <v-form ref="signInForm" v-model="validForm">
                            <v-row class="d-flex align-center" fill>
                                <v-col class="text-center pt-6" cols="12" md="12"
sm="12">
                                    <h3
                                        class="font-italic"
                                        style="background-color: saddlebrown;
color: white"
                                    >
                                        SIGN IN
                                    </h3>
                                </v-col>
                            </v-row>
                            <v-col cols="12" md="12" sm="12" class="pt-3 pb-0">
                                <v-text-field
                                    v-model="loginUser.email"
```

```
                                placeholder="Enter Email"
                                label="Email"
                                color="brown"
                                dense
                                outlined
                                prepend-inner-icon="mdi-email-outline"
                                :rules="[rules.required, rules.email]"
                        />
                </v-col>
                <v-col cols="12" md="12" sm="12" class="py-0">
                        <v-text-field
                                v-model="loginUser.password"
                                placeholder="Enter Password"
                                label="Password"
                                dense
                                outlined
                                color="brown"
                                prepend-inner-icon="mdi-lock-outline"
                                :append-icon="togglePasswordIcon ? 'mdi-eye' :
'mdi-eye-off'"
                                :type="togglePasswordIcon ? 'text' : 'password'"
                                :rules="[rules.required, rules.maxlength]"
                                @click:append="togglePasswordIcon =
!togglePasswordIcon"
                        />
                </v-col>
                <v-col cols="12" md="12" sm="12" class="text-end pt-0">
                        <v-btn color="#DABDAB" @click="onClick"
:disabled="!validForm"
                                >Sign In</v-btn
                        >
                </v-col>
                <v-row class="d-flex align-center">
```

```
                                        <v-col class="text-end" cols="6">Did not register yet?</v-col>
                                        <v-col class="mb-1 pl-0" cols="6">
                                            <v-btn
                                                outlined
                                                color="brown"
                                                class="ml-0"
                                                size="small"
                                                :scrollable="false"
                                                @click="signUpDialog = true"
                                            >
                                                Register Now
                                            </v-btn>
                                        </v-col>
                                    </v-row>
                                </v-form>
                            </v-card>
                        </v-row>
                    </v-container>
                    <v-dialog
                        v-model="signUpDialog"
                        max-width="600px"
                        persistent
                        transition="dialog-bottom-transition"
                    >
                        <v-card>
                            <ModalSignUp ref="modalSignUp" @onClose="onModalClose" />
                        </v-card>
                    </v-dialog>

                    <v-footer>
                        <v-card
```

```
                                        width="100%"
                                        color="brown"
                                        height="40px"
                                        class="d-flex align-center justify-center"
                                >
                                        &copy; 2022 Bookee - All Rights Reserved
                                </v-card>
                        </v-footer>
                </v-app>
</template>
<script>
import ModalSignUp from '../components/ModalSignUp.vue';
import { mapState, mapActions } from 'vuex';
export default {
        name: 'Home-Page',

        components: {
                ModalSignUp,
        },

        data: () => ({
                validForm: false,
                togglePasswordIcon: false,
                signUpDialog: false,
                loginUser: {
                        email: 'admin@gmail.com',
                        password: 'sajjad734',
                },
                rules: {
                        required: (a) => !!a || 'This field is required',
                        maxlength: (a) => a.length >= 8 || 'Minimum 8 characters are required',
```

```
                        email: (v) => /.+@.+\..+/.test(v) || 'E-mail must be valid',
                },
        }),
        computed: {
                ...mapState('global', ['user']),
        },
        methods: {
                ...mapActions('global', ['login', 'fetchLoggedInUser']),
                async onClick() {
                        await this.login(this.loginUser);
                        try {
                        } catch (e) {
                                this.$toast.error('Invalid Email or Password');
                        }
                        this.$router.push({ name: 'home' });
                        this.$toast.success('Welcome. Happy Reading!');
                },
                onModalClose() {
                        this.cleanSignUpForm();
                        this.signUpDialog = false;
                },
                cleanSignUpForm() {
                        this.$refs.modalSignUp.$refs.form.resetValidation();
                        this.$refs.modalSignUp.$refs.form.reset();
                },
        },
};
</script>
<style scoped>
.bg {
        background: url('../assets/background_new.svg');
```

```
        background-size: 100%;
        height: 100%;
}
</style>
```

HOMEPAGE.VUE

```
<template>
        <div>
                <v-row class="justify-center">
                        <v-col cols="4" class="text-center">
                                <v-row class="pa-3 pb-0">
                                        <v-card width="100%">
                                                <h2>Currently Reading</h2>
                                        </v-card>
                                </v-row>
                                <v-row v-if="!booksReading.length" class="px-3 pt-2">
                                        <v-card width="100%"
                                                ><p style="font-size: 17px" class="mb-0">No Book Added!</p></v-
card
                                        >
                                </v-row>
                                <v-row v-for="(book, index) in booksReading" :key="index">
                                        <v-col cols="12" class="pt-6">
                                                <BookDetails
                                                        :isCurrentBook="true"
                                                        :book="book"
                                                        :user="user"
                                                        @onBookSelection="onBookSelection"
                                                >
                                                        <h3 class="book-title">{{ book.name }}</h3>
                                                </BookDetails>
                                        </v-col>
                                </v-row>
```

```
                    </v-col>
                    <v-col cols="6">
                        <BooksList
                            :booksList="booksList"
                            :searchQuery="searchQuery"
                            @onBookSelection="onBookSelection"
                            @newBookAdded="newBookAdded"
                        />
                    </v-col>
                    <v-col cols="2">
                        <!-- <BookDetails /> -->
                    </v-col>
                </v-row>
            </div>
</template>
<script>
import BookDetails from '../components/BookDetails.vue';
import BooksList from '../components/BooksList.vue';
import { mapState, mapActions } from 'vuex'
export default {
    name: 'Home-Page',
    data: () => ({
        searchQuery: '',
    }),
    components: {
        BookDetails,
        BooksList,
    },
    computed: {
        ...mapState('global', ['books', 'user', 'bookReading']),
        booksList() {
```

```
                return this.books;
            },
            booksReading() {
                const booksArray = this.bookReading.map((book) => {
                    return book.book;
                });
                return booksArray;
            },
        },
        methods: {
            ...mapActions('global', [
                'fetchAllBooks',
                'getBookReading',
                'addUserBook',
                'createBook',
            ]),
            async onBookSelection(data) {
                try {
                    await this.addUserBook(data);
                    const response = 'Book Added to ' + data.status.toUpperCase();
                    this.$toast.success(response);
                    await this.fetchData();
                } catch (e) {
                    this.$toast.error('Book Already Added');
                }
            },
            async newBookAdded(newBook) {
                let response = {};
                try {
                    response = await this.createBook(newBook);
                    await this.fetchAllBooks(true);
```

```
                        this.$toast.success('Book Added');
                    } catch (e) {
                        this.$toast.error('Book Already Added');
                    }
                },
                async fetchData() {
                    await this.fetchAllBooks();
                    await this.getBookReading(this.user.id);
                },
        },
        async mounted() {
                await this.fetchData();
        },
};
</script>
<style>
.book-title {
        font-size: 21px;
}
</style>
```

COMMUNITYPAGE.VUE

```
<template>
        <v-row class="justify-center">
                <v-col cols="2" class="px-1">
                        <MorePeople :allUsers="allUsersList" @userFollowed="userFollowed" />
                </v-col>
                <v-col cols="6">
                        <CommunityList
                                :community-list="communityList"
                                :currentUser="currentUser"
                                @onBookSelection="onBookSelection"
```

```
                                        @userFollowed="userFollowed"
                                        @userUnfollowed="unfollowFriend"
                                />
                        </v-col>
                        <v-col cols="4">
                                <FriendsList
                                        :friendsList="friendsList"
                                        :user="currentUser"
                                        @unFollow="unfollowFriend"
                                />
                        </v-col>
                </v-row>
</template>


<script>
import CommunityList from '../components/CommunityList';
import FriendsList from '../components/FriendsList';
import MorePeople from '../components/MorePeople.vue';
import { mapState, mapActions } from 'vuex';
export default {
        name: 'CommunityPage',
        components: {
                CommunityList,
                FriendsList,
                MorePeople,
        },
        computed: {
                ...mapState('global', ['community', 'user', 'friends', 'allUsers']),
                currentUser() {
                        return this.user;
                },
```

```
            friendsList() {
                return this.friends;
            },
            communityList() {
                return this.community;
            },
            allUsersList() {
                return this.allUsers;
            },
        },
        methods: {
            ...mapActions('global', [
                'getMe',
                'fetchCommunity',
                'getAllFriends',
                'removeFriend',
                'addUserBook',
                'addNewFriend',
                'fetchUsersData',
            ]),
            async unfollowFriend(data) {
                await this.removeFriend(data);
                try {
                    this.$toast.success('User UnFollowed');
                    await this.fetchData();
                } catch (e) {
                    this.$toast.error('User Not Found');
                }
            },
            async userFollowed(data) {
                if (!data.user) {
```

```
                        data = {
                                friend: data.friend,
                                user: this.currentUser,
                        };
                }
                try {
                        const isAdded = await this.addNewFriend(data);
                        if (!isAdded) {
                                return this.$toast.error('User Followed Already');
                        }
                        this.$toast.success('User Followed');
                        // this.currentUser.friends = this.currentUser.friends + data.friendId;
                        // await this.getAllFriends(this.currentUser.id);
                        await this.fetchData();
                } catch (e) {
                        this.$toast.error('User Already Followed');
                }
        },
        async onBookSelection(data) {
                try {
                        await this.addUserBook(data);
                        const response = 'Book Added to ' + data.status.toUpperCase();
                        this.$toast.success(response);
                        this.fetchCommunity();
                } catch (e) {
                        this.$toast.error('Book Already Added');
                }
        },
        async fetchData() {
                const user = await this.getMe();
                await this.fetchCommunity();
```

```
                        await this.getAllFriends(user.id);
                        await this.fetchUsersData(false);
                    },
            },
            async created() {
                    await this.fetchData();
            },
};
</script>
<style></style>
```

LAYOUTPAGE.VUE

```
<template>
  <div>
    <NavBar />
    <v-container
      fluid
      class="px-16"
      ref="nav-bar"
    >
      <router-view />
    </v-container>
  </div>
</template>

<script>
import NavBar from '@/components/NavBar';

export default {
  name: 'LayoutPage',
  components: {
    NavBar,
```

```
  },
};
</script>
```

BACKEND
CONTROLLERS/INDEX.JS

```
 ("use strict");
const { asyncMiddleware } = global;
const { ROLES, STATUS } = global.appEnums;
const { getHashedPassword } = global.commonFunctions;
const { Users, UserBooks, Books } = global.db;
const SendMAil = require(`${global.paths.lib}/email-sender`);
const { sequelizeConfig } = require(`${global.paths.lib}/sequelize`);
const { decodeAPiToken } = global.commonFunctions;
const { Op, Sequelize } = require("sequelize");
const user = require("./user");
const {
  newAndConfirmPasswordValidator,
} = require(`${global.paths.middlewares}/password`);
const BookData = require("../../../BookData.json");
const { upperCase } = require("lodash");
module.exports = (router) => {
  router.post(
    "/login",
    asyncMiddleware(async (req, res) => {
      const { email, password } = req.body;
      const user = await Users.$$findOne({
        query: { where: { email, password: getHashedPassword(password) } },
        error: "Invalid email or password!",
      });
      res.http200({
        token: user.createToken(),
```

```
      user,
    });
  })
);
router.post(
  "/signup",
  asyncMiddleware(async (req, res) => {
    const user = await Users.create({ ...req.body, role: ROLES.reader });
    res.http200({ token: user.createToken(), user });
  })
);
router.get(
  "/me",
  asyncMiddleware(async (req, res) => {
    res.http200({
      user: req.user,
    });
  })
);
router.get(
  "/all-users",
  asyncMiddleware(async (req, res) => {
    const query = {
      where: {},
      order: sequelizeConfig.Order.Desc(),
      attributes: [
        "id",
        "firstName",
        "lastName",
        "initials",
        "fullName",
```

```
        "email",
        "friends",
      ],
    };
    const { rows, count } = await Users.findAndCountAll(query);
    const users = rows.map((obj, index, self) => {
      obj.dataValues = {
        ...obj.dataValues,
        friendsList: [],
      };
      self.forEach((e) => {
        const u = e.dataValues;
        if (obj.dataValues.friends && obj.dataValues.friends.includes(u.id)) {
          obj.dataValues.friendsList.push(u);
        }
      });
      return obj;
    });
    res.http200(users, { count: count });
  })
);
router.post(
  "/verify-token",
  asyncMiddleware(async (req, res) => {
    const { token } = req.body;
    const user = await getUserFromToken(token);
    if (token == user.token) {
      return res.http200("Token verified");
    }
    return res.http500("Token unverified");
  })
```

```
);
router.put(
  "/reset-password",
  [newAndConfirmPasswordValidator],
  asyncMiddleware(async (req, res) => {
    const { token, confirmPassword } = req.body;
    const user = await getUserFromToken(token);

    const updatedUser = await user.update({
      password: confirmPassword,
      token: "",
    });
    return res.http200(updatedUser);
  })
);
router.post(
  "/add-book",
  asyncMiddleware(async (req, res) => {
    const { userId, bookId, isReading, isComplete, inQueue, status } =
      req.body;
    let bookStatus = "";
    if (!status) {
      if (isReading) {
        bookStatus = STATUS.Book.reading;
      } else if (isComplete) {
        bookStatus = STATUS.Book.complete;
      } else {
        bookStatus = STATUS.Book.inQueue;
      }
    } else {
      bookStatus = status;
```

```
    }
    const [userBook, created] = await UserBooks.findOrCreate({
      where: {
        userId,
        bookId,
      },
      defaults: {
        userId: userId,
        bookId: bookId,
        status: bookStatus,
      },
    });
    if (!created && userBook.status !== bookStatus) {
      const updatedBook = userBook.update({
        status: bookStatus,
      });
    } else if (!created) {
      res.http500("Book Already Added");
    }
    // console.log('user-book ===== 2', userBook);
    res.http200(userBook);
  })
);


router.route("/all-books").get(
  asyncMiddleware(async (req, res) => {
    const query = {
      where: {},
      order: sequelizeConfig.Order.Desc(),
    };
    const books = await Books.findAndCountAll(query);
```

```
    return res.http200(books.rows, { count: books.count });
  })
);
router.param(
  "userId",
  asyncMiddleware(async (req, res, next, userId) => {
    const user = await Users.$$findByPk({ id: +userId });
    req.user = user;
    next();
  })
);
router.get(
  "/community",
  asyncMiddleware(async (req, res) => {
    const { count, rows } = await UserBooks.findAndCountAll({
      order: sequelizeConfig.Order.Desc(),
      include: [
        {
          model: Books,
          as: Books.$$singularName,
        },
        {
          model: Users,
          as: Users.$$singularName,
        },
      ],
    });
    return res.http200(rows, {
      count: count,
    });
  })
```

```
);
router.get(
  "/user-books/:userId",
  asyncMiddleware(async (req, res) => {
    const userId = req.params.userId;
    const userBooks = await UserBooks.findAndCountAll({
      where: {
        userId,
      },
      include: [
        {
          model: Books,
          as: Books.$$singularName,
        },
      ],
    });
    return res.http200(userBooks.rows, { count: userBooks.count });
  })
);
router.get(
  "/all-friends/:userId",
  asyncMiddleware(async (req, res) => {
    if (!req.user.friends) {
      return res.http200("Friends Not Found", { count: 0 });
    }
    const splittedIds = req.user.friends.split(",");
    let friendIds = splittedIds.map((id) => +id).filter((id) => !!id);
    let { rows, count } = await UserBooks.findAndCountAll({
      where: {
        userId: {
          [Op.or]: friendIds,
```

```
        },
      },
      include: [
        {
          model: Books,
          as: Books.$$singularName,
        },
        {
          model: Users,
          as: Users.$$singularName,
          attributes: [
            "id",
            "firstName",
            "lastName",
            "initials",
            "fullName",
            "email",
            "friends",
          ],
        },
      ],
    });
    const friends = rows
      .map((obj) => obj.dataValues)
      .filter(
        (obj, index, self) =>
          index === self.findIndex((t) => t.userId === obj.userId)
      );
    friendIds = friendIds.filter(
      (id) => friends.findIndex((obj) => obj.userId === id) <= -1
    );
```

```
    console.log(friendIds, "ajj");
    let allFriends = friends;
    if (friendIds.length) {
      const users = await Users.findAll({
        where: {
          id: {
            [Op.or]: friendIds,
          },
        },
        attributes: [
          "id",
          "firstName",
          "lastName",
          "initials",
          "fullName",
          "email",
          "friends",
        ],
      });
      allFriends = [
        ...users.map((obj) => {
          return {
            ...obj.dataValues,
            initials: upperCase(
              `${obj.dataValues.firstName[0]}${obj.dataValues.lastName[0]}`
            ),
            fullName: `${obj.dataValues.firstName} ${obj.dataValues.lastName}`,
          };
        }),
        ...friends,
      ];
```

```
    }
    return res.http200(allFriends, {
      count: allFriends.length,
    });
  })
);
router.get(
  "/book-reading/:userId",
  asyncMiddleware(async (req, res) => {
    const userId = req.params.userId;
    const bookReading = await UserBooks.findAndCountAll({
      where: {
        userId,
        status: STATUS.Book.reading,
      },
      include: [
        {
          model: Books,
          as: Books.$$singularName,
        },
      ],
    });
    return res.http200(bookReading.rows, { count: bookReading.count });
  })
);

router.put(
  "/remove-friend/:userId",
  asyncMiddleware(async (req, res) => {
    const { userId, friendId } = req.body;
    const user = req.user;
```

```
    if (!user.friends || !user.friends.includes(friendId)) {
      return res.http500("No Friend Found");
    }
    if (user.friends.length <= 2) {
      user.friends = "";
    } else {
      const updatedUserFriends = user.friends.replace(friendId, "");
      user.friends = updatedUserFriends;
    }
    // const friends = user.friends;
    // console.log("user.friends====>>", user.friends);
    const response = user.update({ friends: user.friends });
    return res.http200(user);
  })
);
router.put(
  "/add-friend/:userId",
  asyncMiddleware(async (req, res) => {
    const { userId, friendId } = req.body;

    if (req.user.friends?.includes(friendId)) {
      return res.http500("Friend Already Added");
    }
    const friend = await Users.$$findByPk({ id: +friendId });
    if (!friend) {
      return res.http500("Friend Id Invalid");
    }
    let updatedFriends = "";
    if (!req.user.friends?.length) {
      updatedFriends = updatedFriends + friendId;
    } else {
```

```
      updatedFriends = req.user.friends + "," + friendId;
    }
    const response = req.user.update({ friends: updatedFriends });
    return res.http200(req.user);
  })
 );
};
async function getUserFromToken(token) {
 const decodedToken = await decodeAPiToken(token);
 const user = await Users.$$findOne({
  query: {
   where: {
    email: decodedToken.email,
   },
  },
 });
 return user;
}
```

CONTROLLERS/BOOKS.JS

```
const { asyncMiddleware } = global;
const { sequelizeConfig } = require(`${global.paths.lib}/sequelize`);
const { Books, Sequelize } = global.db;

module.exports = (router) => {
 router
  .route("/")
  .get(
   asyncMiddleware(async (req, res) => {
    const query = {
     where: {},
     order: sequelizeConfig.Order.Desc(),
```

```
      };
      const books = await Books.findAndCountAll(query);
      return res.http200(books.rows, { count: books.count });
    })
  )
  .post(
    asyncMiddleware(async (req, res) => {
      delete req.body.id;
      const record = await Books.findOne({
        where: {
          name: req.body.name,
          author: req.body.author,
          description: req.body.description,
          genre: req.body.genre,
        },
      });
      if (record) {
        return res.http500("Enter Unique Book");
      }
      const books = await Books.create(req.body);
      return res.http200(books);
    })
  );


router.param(
  "bookId",
  asyncMiddleware(async (req, res, next, bookId) => {
    const book = await Books.$$findByPk({
      id: +bookId,
    });
    req.book = book;
```

```
    next();
  })
);

router
  .route("/:bookId")
  .get(
    asyncMiddleware(async (req, res) => {
      res.http200(req.book);
    })
  )
  .put(
    asyncMiddleware(async (req, res) => {
      const book = req.body;
      const updatedbook = await req.book.update(book);
      return res.http200(updatedbook);
    })
  )
  .delete(
    asyncMiddleware(async (req, res) => {
      await Books.destroy({ where: { id: req.params.bookId } });
      return res.http200("Deleted book successfully!");
    })
  );
};
```

Danish Akram Gondal 2021-CS-123
M Jawad Haider 2021-CS-149