TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO PROJECT 3

App mobile quản lí học tập cho sinh viên và giảng viên

TĂNG MINH VŨ
Vu.tm215519@sis.hust.edu.vn
MSSV: 20215519

Ngành Khoa Học Máy Tính

Giảng viên hướng dẫn: Thầy Nguyễn Thanh Hùng

ĐỀ TÀI PROJECT3

Tóm tắt nội dung đồ án

Dự án là một ứng dụng di động được phát triển trên nền tảng React Native, nhằm phục vụ việc quản lý lớp học cho sinh viên và giảng viên. App mobile cung cấp giải pháp quản lí lớp học, đăng bài giảng, bài tập, điểm danh cho giảng viên và giải pháp đăng kí lớp học, làm bài tập, xin nghỉ học, nhận các thông báo mới nhất từ giảng viên cho sinh viên, và hệ thống nhắn tin realtime cho cả giảng viên và sinh viên.

MỤC LỤC

CHƯƠNG 1. Quy trình thực hiện

1.1 Khởi tạo dự án React Native

Dự án được khởi tạo sử dụng Expo với cấu hình như sau:

```json
{
  "name": "qldt",
  "main": "expo-router/entry",
  "version": "1.0.0",
  "scripts": {
    "start": "expo start",
    "android": "expo run:android",
    "ios": "expo run:ios",
    "web": "expo start --web"
  }
}
```

Các dependencies chính:

- expo-router: Quản lý navigation
- react-native-paper: UI components
- @reduxjs/toolkit: State management
- axios: HTTP client
- react-native-reanimated: Animations
- react-native-push-notification: Thông báo

1.2 Xây dựng UI

UI được xây dựng theo nguyên tắc component-based với các thành phần chính:

- ThemedComponents: Components có thể thay đổi theme
- CustomComponents: Components tùy chỉnh cho ứng dụng
- Layouts: Các layout chung cho các màn hình
- Screens: Các màn hình chính của ứng dụng

```
function NotificationItem(props: ClassProps) {
  return (
    <ThemedView style={styles.container}>
      <TouchableRipple onPress={detailView}>
        <View style={styles.content}>
          <Text style={styles.title}>{props.title}</Text>
          <Text style={styles.description}>{props.content}</Text>
        </View>
      </TouchableRipple>
    </ThemedView>
  );
}
```

1.3 Quản lý state và navigation

State management:

- Redux được sử dụng để quản lý global state
- Async Storage lưu trữ local data
- Context API cho state chia sẻ giữa các components

Navigation:

- Expo Router quản lý navigation với file-based routing
- Stack và Tab navigation cho luồng di chuyển
- Deep linking cho notification và sharing

1.4 Tích hợp API và xử lý dữ liệu

API được tổ chức thành các service riêng biệt:

- AuthApi: Xử lý xác thực và phân quyền

```
class AuthApi {
    async login(email, password, deviceId, fcm_token) {
        const response = await axios.post(`${base_api}/it4788/login`, {
            email, password, device_id: deviceId, fcm_token
        });
        return response;
    }

    async signup(ho, ten, email, password, uuid, role) {
        const response = await axios.post(`${base_api}/it4788/signup`, {
            ho, ten, email, password, uuid, role
        });
        return response;
    }
}
```

- ClassApi: Quản lý lớp học và điểm danh

```
class ClassApi {
    async getClassList(token, role, accountId) {
        return await axios.post(`${base_api}/it5023e/get_class_list`, {
            token, role, account_id: accountId
        });
    }

    async takeAttendance(token, classId, date, attendanceList) {
        return await axios.post(`${base_api}/it5023e/take_attendance`, {
            token, class_id: classId, date, attendance_list: attendanceList
        });
    }
}
```

- AssignmentApi: Quản lý bài tập và nộp bài

```
class AssignmentApi {
    async createSurvey(formData) {
        const response = await axios.post(
            `${base_api}/it5023e/create_survey`,
            formData,
            {
                headers: { 'Content-Type': 'multipart/form-data' }
            }
        );
        return response;
    }

    async submitSurvey(formData) {
        return await axios.post(
            `${base_api}/it5023e/submit_survey`,
            formData
        );
    }
}
```

1.5 Xác thực và Phân quyền

Hệ thống sử dụng token-based authentication:

1. Đăng nhập:
   - Gửi thông tin đăng nhập (email/password)
   - Nhận JWT token từ server
   - Lưu token vào AsyncStorage
2. Phân quyền:
   - Phân chia route theo role (student/teacher)
   - Kiểm tra quyền trước khi thực hiện các chức năng
   - Hiển thị UI tương ứng với role
3. Refresh token:

- Tự động làm mới token khi hết hạn
- Đăng xuất khi refresh token thất bại

1.6 Push Notification

Notification được xử lý qua các bước:

1. Đăng ký device token
2. Xử lý notification

```
Notifications.setNotificationHandler({
    handleNotification: async (notification) => {
        playSound();
        if (notification.request?.content?.data?.type == "NOTIFICATION") {
            store.dispatch(updateUnreadNotifiCount(1));
        }
        return {
            shouldShowAlert: true,
            shouldPlaySound: true,
            shouldSetBadge: false,
        };
    }
});
```

3. Lưu trữ và hiển thị thông báo trong app

1.7 Offline Mode và Caching

Ứng dụng được thiết kế để hoạt động khi không có kết nối mạng:

1. Kiểm tra kết nối:

```
export const checkConnectivity = async () => {
    const netInfo = await NetInfo.fetch();
    return netInfo.isConnected && netInfo.isInternetReachable;
};
```

2. Lưu trữ dữ liệu offline:

```
export const saveOfflineData = async (key, data) => {
    try {
        await AsyncStorage.setItem(key, JSON.stringify(data));
    } catch (error) {
        console.error('Error saving offline data:', error);
    }
};
```

3. Đồng bộ hóa khi có mạng:

- Tự động đồng bộ dữ liệu khi có kết nối
- Xử lý xung đột dữ liệu

- Thông báo cho người dùng

1.8 Data Storage

Dữ liệu được lưu trữ ở nhiều cấp độ:

1. Redux Store:
- Lưu trữ trạng thái toàn cục
- Quản lý data cho UI
- Caching tạm thời
2. AsyncStorage:
- Lưu token và thông tin user
- Cache dữ liệu offline
- Cấu hình người dùng
3. File Storage:
- Lưu file tải về
- Cache hình ảnh
- Tài liệu tạm thời

1.9 Infrastructure Setup

1. Development Environment:

```
{
  "development": {
    "developmentClient": true,
    "distribution": "internal"
  }
}
```

2. Production Environment:

```
{
  "production": {
    "autoIncrement": true,
    "android": {
      "buildType": "apk"
    }
  }
}
```

3. CI/CD Pipeline:
- Automated testing
- Build automation
- Continuous deployment

1.10 Development Workflow

1. Version Control:

- Git flow workflow
- Feature branches
- Pull request review

2. Code Quality:

- ESLint configuration
- TypeScript type checking
- Unit testing

3. Documentation:

- Code comments
- API documentation
- User guide

1.11 Những Thách Thức Chính

1. Performance Optimization:

- Tối ưu render cycle của React Native
- Xử lý danh sách dài (virtualization)
- Quản lý bộ nhớ và cache

2. Cross-platform Compatibility:

- UI khác biệt giữa iOS và Android
- Xử lý quyền truy cập thiết bị
- Push notification trên các nền tảng

3. Offline Support:

- Đồng bộ dữ liệu hai chiều
- Xử lý xung đột khi offline
- Quản lý storage giới hạn

4. Độ ổn định:

- Xử lý lỗi mạng
- Recovery mechanism
- Error boundary

CHƯƠNG 2. Sơ đồ hệ thống

2.1 Sơ đồ tổng quan

Hệ thống được chia thành các module chính:

1. Authentication Module:
- Login/Register
- Password Recovery
- Session Management
2. Class Management Module:
- Class Registration
- Schedule View
- Attendance
3. Assignment Module:
- Assignment Creation
- Submission
- Grading
4. Communication Module:
- Chat
- Notifications
- File Sharing
5. Administration Module:
- User Management
- Permission Control
- System Settings

2.2 Luồng xử lý chính

2.2.1 Authentication Flow

1. Đăng nhập:

```
const onLoginPressed = async () => {
    // Validate input
    const emailError = emailValidator(email.value);
    const passwordError = passwordValidator(password.value);

    // Call login API
    const response = await authApi.login(email, password, deviceId, fcm_token);

    // Handle response
    if (response.data.code === "1000") {
        await SyncStorage.set("token", userData.token);
        await SyncStorage.set("user_info", JSON.stringify(userData));
        router.replace("(students)");
    }
}
```

2. Đăng ký:

```
const onSignUpPressed = async () => {
    // Validate input fields
    const response = await authApi.signup(firstName, name, email, password, uuid, role);

    // Send verification code
    const verifyCode = response.data.verify_code;
    router.push({
        pathname: 'VerifyEmail',
        params: { email, password, verify_code: verifyCode }
    });
}
```

2.2.2 Class Registration Flow

1. Xem danh sách lớp:

```
const fetchClasses = async () => {
    const token = await SyncStorage.get('token');
    const response = await classApi.getClassList(token, role, userId);

    if (response.data?.meta?.code === "1000") {
        const allClasses = response.data.data?.page_content || [];
        setClasses(allClasses);
    }
}
```

2. Đăng ký lớp:

```
const handleRegisterClass = async () => {
    const classIds = listRegisterClass.map(item => item.classCode);
    const response = await classApi.registerClass(token, classIds);

    if (response.data?.meta?.code === "1000") {
        ToastAndroid.show("Đăng ký lớp thành công", ToastAndroid.SHORT);
        await fetchClassList();
    }
}
```

2.2.3 Assignment Flow

1. Tạo bài tập (Giảng viên):

```
const handleCreateAssignment = async () => {
    const formData = new FormData();
    formData.append('token', token);
    formData.append('classId', classId);
    formData.append('title', assignmentName);
    formData.append('deadline', endDate.toISOString());
    formData.append('file', {
        uri: file.uri,
        type: file.mimeType,
        name: file.name
    });

    const response = await assignmentApi.createSurvey(formData);
    if (response.data?.meta?.code === "1000") {
        ToastAndroid.show("Tạo bài tập thành công", ToastAndroid.SHORT);
    }
}
```

2. Nộp bài tập (Sinh viên):

```
const handleSubmit = async () => {
    const formData = new FormData();
    formData.append('token', token);
    formData.append('assignmentId', id);
    formData.append('textResponse', answer.value);
    formData.append('file', {
        uri: file.uri,
        type: file.mimeType,
        name: file.name
    });

    const response = await assignmentApi.submitSurvey(formData);
    if (response.data?.meta?.code === "1000") {
        ToastAndroid.show("Nộp bài thành công", ToastAndroid.SHORT);
    }
}
```

2.2.4 Attendance Flow

1. Điểm danh (Giảng viên):

```
const handleTakeAttendance = async () => {
    try {
        const token = await SyncStorage.get('token');
        const formattedDate = format(selectedDate, 'yyyy-MM-dd');

        // Thực hiện điểm danh
        const response = await classApi.takeAttendance(
            token,
            classId,
            formattedDate,
            attendanceList
        );

        if (response.data?.meta?.code === "1000") {
            ToastAndroid.show("Điểm danh thành công", ToastAndroid.SHORT);
            await loadAttendanceList();
        }
    } catch (error) {
        console.error('Take attendance error:', error);
        ToastAndroid.show("Lỗi điểm danh", ToastAndroid.SHORT);
    }
}
```

2.2.5 Absence Request Flow

1. Gửi đơn xin nghỉ (Sinh viên):

```
const handleCreateAbsence = async () => {
    const formData = new FormData();
    formData.append('token', token);
    formData.append('classId', classId);
    formData.append('date', date.toISOString().split('T')[0]);
    formData.append('reason', description.value);
    formData.append('title', title.value);

    if (file) {
        formData.append('file', {
            uri: file.uri,
            type: file.mimeType,
            name: file.name
        });
    }

    const response = await classApi.requestAbsence(formData);
    if (response.data?.meta?.code === "1000") {
        // Gửi thông báo cho giảng viên
        await notificationApi.sendNotification(
            token,
            `Sinh viên ${account?.name} xin nghỉ học`,
```

```
            parseInt(response.data.data.page_content[0].account_id),
            "ABSENCE"
        );
    }
}
```

1. Duyệt đơn nghỉ (Giảng viên):

```
const handleReviewRequest = async (requestId, status) => {
    try {
        const token = await SyncStorage.get('token');
        const response = await classApi.reviewAbsenceRequest(
            token,
            requestId,
            status
        );

        if (response.data?.meta?.code === "1000") {
            // Gửi thông báo cho sinh viên
            await notificationApi.sendNotification(
                token,
                `Đơn xin nghỉ của bạn đã ${status === 'ACCEPTED' ? 'được chấp nhận' : 'bị từ chối'}`,
                absenceRequest.student_id,
                status === 'ACCEPTED' ? 'ACCEPT_ABSENCE_REQUEST' : 'REJECT_ABSENCE_REQUEST'
            );

            ToastAndroid.show(
                status === 'ACCEPTED' ? "Đã duyệt đơn" : "Đã từ chối đơn",
                ToastAndroid.SHORT
            );
        }
    } catch (error) {
        console.error('Review absence request error:', error);
    }
}
```

2.3 Kiến trúc và công nghệ

2.3.1 Frontend Architecture

1. Component Layer:
- UI Components
- Screen Components
- Navigation Components
2. Business Logic Layer:
- Redux Actions/Reducers
- Service Classes
- Utils & Helpers
3. Data Layer:
- API Clients
- Local Storage
- File System

2.3.2 State Management Structure

1. Redux Store:

```
// Store Configuration
const store = configureStore({
  reducer: {
    userInfo: userInfoReducer,
    conversations: conversationsReducer,
    unreadNotifiCount: notificationReducer,
    newMessageCount: messageCountReducer
  }
});
```

2. Local Storage System:

```
// Storage Utils
export const saveUserInfo = async (userData) => {
    try {
        await AsyncStorage.setItem('user_info', JSON.stringify(userData));
        await SyncStorage.set('token', userData.token);
        await SyncStorage.set('role', userData.role);
    } catch (error) {
        console.error('Save user info error:', error);
    }
};
```

2.3.3 Network Layer

1. API Client:

```
class BaseApi {
    async makeRequest(endpoint, method, data, headers = {}) {
        try {
            const response = await axios({
                url: `${base_api}${endpoint}`,
                method,
                data,
                headers: {
                    'Content-Type': 'application/json',
                    ...headers
                }
            });
            return response;
        } catch (error) {
            this.handleError(error);
        }
    }

    handleError(error) {
        if (error.response?.status === 401) {
            // Handle authentication error
        }
        throw error;
    }
}
```

2. WebSocket Connection:

```
export const connectSocket = (userId) => {
    const client = new Client({
        brokerURL: WEBSOCKET_URL,
        connectHeaders: {
            login: 'guest',
            passcode: 'guest'
        },
        debug: function (str) {
            console.log(`STOMP: ${str}`);
        }
    });
```

```
    client.onConnect = () => {
        client.subscribe(`/user/${userId}/queue/messages`, onMessageReceived);
    };

    client.activate();
};
```

CHƯƠNG 3. Tính năng của hệ thống

3.1 Quản lý lớp học

1. Quản lý danh sách lớp:

```
// Phân loại lớp học theo trạng thái
const classesData = allClasses.filter(c =>
    (c.class_type === 'LT' || c.class_type === 'LT_BT') &&
    !isClassExpired(c.end_date)
) || [];

const groupsData = allClasses.filter(c =>
    c.class_type === 'BT' &&
    !isClassExpired(c.end_date)
) || [];

const expiredData = allClasses.filter(c =>
    isClassExpired(c.end_date)
) || [];
```

2. Tạo lớp học mới (Giảng viên):

```
const handleCreateClass = async () => {
    const response = await classApi.createClass(
        token,
        classCode.value,
        className.value,
        classType.value,
        startDate,
        endDate,
        maxStudent.value
    );
```

```
    if (response.data?.meta?.code === "1000") {
        ToastAndroid.show("Tạo lớp thành công", ToastAndroid.SHORT);
        router.push("/(teachers)/manager_class");
    }
}
```

3. Quản lý thông tin lớp:

```
// Chỉnh sửa thông tin lớp
const handleFixClass = async () => {
    const response = await classApi.editClass(
        token,
        classid,
        className.value,
        type.value,
        startDate.toISOString().split('T')[0],
        endDate.toISOString().split('T')[0]
    );

    if (response.data?.meta?.code === "1000") {
        ToastAndroid.show("Cập nhật lớp học thành công", ToastAndroid.SHORT);
        router.back();
    }
}
```

3.2 Quản lý bài tập

1. Hiển thị danh sách bài tập:

```
const fetchAssignments = async () => {
    const token = await SyncStorage.get('token');
    if (role === 'LECTURER') {
        const response = await assignmentApi.getAllSurveys(token, classId);
        setAssignments(response.data.data || []);
    } else {
        const response = await assignmentApi.getStudentAssignments(token);
        const formattedAssignments = response.data.data.map(assignment => ({
            ...assignment,
            status: getAssignmentStatus(assignment)
        }));
        setAssignments(formattedAssignments);
    }
}
```

2. Xem chi tiết và chấm điểm:

```
const handleGradeSubmission = async () => {
    const grade = {
        score: gradeNum,
        submissionId: selectedItem.id
    };

    const response = await assignmentApi.getSurveyResponse(
        token,
        id,
        grade
    );

    if (response.data?.meta?.code === "1000") {
        // Gửi thông báo cho sinh viên
        await notificationApi.sendNotification(
            token,
            `Bài tập của bạn đã được chấm điểm: ${gradeNum}điểm`,
            selectedItem.student_account.account_id,
            'ASSIGNMENT_GRADE'
        );
    }
}
```

3. File handling:

```
const handleDocumentPick = async () => {
    const result = await DocumentPicker.getDocumentAsync({
        type: '*/*',
        copyToCacheDirectory: true
    });

    if (!result.canceled) {
        // Validate file size
        if (result.assets[0].size > 10 * 1024 * 1024) {
            ToastAndroid.show(
                "File không được vượt quá 10MB",
                ToastAndroid.SHORT
            );
            return;
        }
        setFile(result.assets[0]);
    }
}
```

3.3 Điểm danh và quản lý vắng mặt

1. Giảng viên thực hiện điểm danh:

```
const TakeAttendance = () => {
    const [attendanceList, setAttendanceList] = useState([]);
    const [selectedDate, setSelectedDate] = useState(new Date());

    // Check attendance record for selected date
    const checkAttendanceRecord = async () => {
        const formattedDate = format(selectedDate, 'yyyy-MM-dd');
        const response = await classApi.getAttendanceList(
            token,
            classId,
            formattedDate
        );

        if (response.data?.meta?.code === "1000") {

setAttendanceList(response.data.data.attendance_student_details);
        }
    }

    // Update student status
    const updateAttendanceStatus = async (studentId, newStatus) => {
        const response = await classApi.setAttendanceStatus(
            token,
            newStatus,
            attendanceRecord.attendanceId
        );

        if (newStatus === "UNEXCUSED_ABSENCE") {
            // Notify student
            await notificationApi.sendNotification(
                token,
                "Bạn bị điểm danh vắng học",
                parseInt(account_id),
                "ABSENCE"
            );
        }
    }
}
```

2. Sinh viên xin nghỉ phép:

```
const AbsenceRequest = () => {
    // Submit absence request
    const handleCreateAbsence = async () => {
        const formData = new FormData();
        formData.append('token', token);
        formData.append('classId', classId);
        formData.append('date', date.toISOString().split('T')[0]);
        formData.append('reason', description.value);

        // Handle supporting documents
        if (file) {
            formData.append('file', {
                uri: file.uri,
```

```
                type: file.mimeType,
                name: file.name
            });
        }

        const response = await classApi.requestAbsence(formData);

        // Notify lecturer
        if (response.data?.meta?.code === "1000") {
            await notificationApi.sendNotification(
                token,
                `Sinh viên ${account?.name} xin nghỉ học trong lớp
${className}`,
                parseInt(lecture_id),
                "ABSENCE"
            );
        }
    }
}
```

3.4 Notifications và Chat

1. Quản lý thông báo:

javascript
Copy

```
// Notification handler
Notifications.setNotificationHandler({
    handleNotification: async (notification) => {
        // Play notification sound
        playSound();

        // Update unread count
        if (notification.request?.content?.data?.type ==
"NOTIFICATION") {
            store.dispatch(updateUnreadNotifiCount(1));
        } else {
            store.dispatch(updateNewMessageCount(1));
        }

        return {
            shouldShowAlert: true,
            shouldPlaySound: true,
            shouldSetBadge: false,
        };
    }
});

// Fetch notifications
const fetchNotifications = async () => {
    const token = await SyncStorage.get('token');
    const response = await notificationApi.getNotifications(token, 0,
100);
```

```
    if (response.data?.meta?.code === "1000") {
        const unreadCount = response.data.data.filter(
            item => item.status === "UNREAD"
        ).length;
        dispatch(setUnreadNotifiCount(unreadCount));
    }
}
```

2. Chat system:

```
// Message handling
const handleSendMessage = async (content = newMessage, isImage =
false) => {
    const user = await JSON.parse(SyncStorage.get("user_info"));

    const messageToSend = {
        receiver: { id: partnerId },
        content: isImage ? `[IMG]${content}[/IMG]` : content,
        sender: user.email,
        token: user.token
    };

    try {
        // Send via WebSocket
        sendMessage(messageToSend);
        setNewMessage("");
    } catch (error) {
        console.error("Error sending message:", error);
    }
};

// Image handling in chat
const pickImage = async () => {
    const result = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.Images,
        base64: true,
        quality: 0.8,
    });

    if (!result.canceled) {
        const base64Data =
`${result.assets[0].base64}`;
```

```
            await sendMessage_(response.data.secure_url, true);
        }
};
```

3.5 Offline Support và Sync

1. Network Status Monitoring

```
const NetworkManager = {
    // Kiểm tra kết nối
    checkConnectivity: async () => {
        const netInfo = await NetInfo.fetch();
        return netInfo.isConnected && netInfo.isInternetReachable;
    },

    // Lưu dữ liệu offline
    saveOfflineData: async (key, data) => {
        try {
            await AsyncStorage.setItem(key, JSON.stringify(data));
        } catch (error) {
            console.error('Error saving offline data:', error);
        }
    },

    // Đồng bộ khi có mạng
    syncOfflineData: async () => {
        const offlineData = await getOfflineData(OFFLINE_KEYS.CLASS_LIST);
        if (offlineData) {
            await uploadOfflineChanges(offlineData);
        }
    }
};
```

2. File Management:

```
const FileManager = {
    // Xử lý file upload
    handleDocumentPick: async () => {
        try {
            const result = await DocumentPicker.getDocumentAsync({
                type: '*/*',
                copyToCacheDirectory: true
            });

            if (!result.canceled) {
                const selectedFile = result.assets[0];
                // Kiểm tra kích thước file
                if (selectedFile.size > 10 * 1024 * 1024) {
                    ToastAndroid.show("File quá lớn", ToastAndroid.SHORT);
                    return null;
                }
                return selectedFile;
            }
        } catch (err) {
            console.error('Document picker error:', err);
            return null;
        }
    },

    // Cache files
    cacheFile: async (fileUri, fileName) => {
```

3. Error Handling System:

```javascript
const ErrorBoundary = {
    // Global error handler
    handleError: (error, componentStack) => {
        console.error('Application Error:', error);
        console.error('Component Stack:', componentStack);

        // Log to monitoring service
        logErrorToService(error, componentStack);

        // Show user friendly message
        ToastAndroid.show(
            "Đã có lỗi xảy ra. Vui lòng thử lại sau.",
            ToastAndroid.LONG
        );
    },

    // Network error handler
    handleNetworkError: async (error) => {
        if (error.response?.status === 401) {
            await handleLogout();
            router.replace('(auths)');
        }
    }
};
```

CHƯƠNG 4. Backend Architecture

4.1.1 Cấu trúc Backend

1. Framework và ngôn ngữ:

- Java 17
- Spring Boot 3.x
- Maven/Gradle build tool

2. Kiến trúc backend:

├── config/ *// Cấu hình hệ thống*
├── controller/ *// REST endpoints*
├── model/ *// Entities & DTOs*
├── repository/ *// Data access layer*
├── service/ *// Business logic*
├── security/ *// Authentication & Authorization*
└── utils/ *// Helper classes*

3. API Endpoints:

@RestController

```
@RequestMapping("/api/v1")
public class ClassController {
   // Quản lý lớp học
   @PostMapping("/class/create")
   @PostMapping("/class/edit")
   @PostMapping("/class/delete")
   @GetMapping("/class/list")

   // Điểm danh
   @PostMapping("/attendance/take")
   @PostMapping("/attendance/update")
   @GetMapping("/attendance/list")

   // Bài tập
   @PostMapping("/assignment/create")
   @PostMapping("/assignment/submit")
   @GetMapping("/assignment/list")
}
```

CHƯƠNG 5. Kết luận và đánh giá

5.1.1 Kết quả đạt được

1. Chức năng chính:
- Quản lý lớp học đầy đủ, phân quyền giảng viên/sinh viên
- Hệ thống bài tập, điểm danh hoạt động ổn định
- Thông báo và chat realtime
- Hỗ trợ offline mode

2. Hiệu năng:
- Load time dưới 2s cho các tác vụ chính
- Sử dụng cache và lazy loading hiệu quả
- Tối ưu size bundle và tài nguyên

3. UX/UI:
- Giao diện thân thiện, dễ sử dụng
- Navigation logic và rõ ràng
- Feedback kịp thời cho người dùng

5.1.2 Hạn chế và hướng phát triển

1. Hạn chế hiện tại:

// Còn một số vấn đề cần cải thiện

- Chưa hỗ trợ video call trong chat
- Chưa có tính năng export dữ liệu
- Đồng bộ offline chưa thật sự mượt mà
- Push notification đôi khi bị delay

2. Hướng phát triển:

// Tính năng cần bổ sung

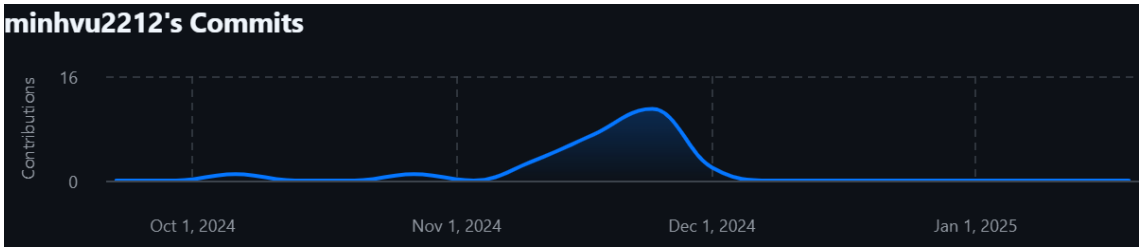
- Tích hợp video call qua WebRTC
- Thêm tính năng thống kê, báo cáo
- Cải thiện UI/UX trên tablet
- Tối ưu hóa push notification
- Tăng cường bảo mật

3. Kế hoạch nâng cấp:

```
const futureUpdates = {
    phase1: "Video call & reporting",
    phase2: "UI/UX improvements",
    phase3: "Performance optimization",
    phase4: "Security enhancements"
}
```

Link Sản phẩm:
https://github.com/minhvu2212/PROJECT3_QLDT_RELEASE_PUBLIC

Proof:




Hoang130203
#1
45 commits 35,112 ++ 4,633 --
Contributions
16
0
Oct 1, 2024
Nov 1, 2024
Dec 1, 2024
Jan 1, 2025
minhvu2212
#2
25 commits 7,888 ++ 3,678 --
Contributions
16
0
Oct 1, 2024
Nov 1, 2024
Dec 1, 2024
Jan 1, 2025
anhvu08903
#3
3 commits 641 ++ 259 --
Contributions
16
0
Oct 1, 2024
Nov 1, 2024
Dec 1, 2024
Jan 1, 2025
pbtruong03
#4
2 commits 1,372 ++ 801 --
Contributions
16
0
Oct 1, 2024
Nov 1, 2024
Dec 1, 2024
Jan 1, 2025
SyNguyenz
#5
2 commits 587 ++ 66 --
Contributions
16
0
Oct 1, 2024
Nov 1, 2024
Dec 1, 2024
Jan 1, 2025