Final Report

NPCs

(Jennifer Duarte, Shambhawi Sharma, Inika Singh, Bridget Torres)

CS487 Software Engineering

12/2/2023

## Updates

In our application BuddyZen we added a share button to the UI sidebar view for calendar sharing on both ends of the users. To develop our application using Android Studio we implemented our pseudo code from the last deliverable. We created the database for our BuddyZen application using Azure. For user authentication, we created a secure login system and integrated Azure to store as well as manage user credentials. We made sure creating events within the application would be seamless using Android Studio and Azure as the backend database to store event data. We implemented the calendar to allow users to be able to see and manage their events by retrieving and displaying data from the Azure database. In addition, we created a user-friendly interface to address password changes in the application and also incorporated secure protocols to update and store password changes in the database. As well as implementing Java code for interaction with JDBC. We also added more UI screens for when an event gets created a screen will pop up prompting the user to fill out the information then it will get added to the calendar. We have screens for tasks that will take the user to another screen to either add tasks, delete tasks, or modify tasks. And then when they click ‘add tasks’ they can choose between assignment, class, or other. When they choose ‘assignment’ they are prompted to another screen where they can fill out information and add it to their calendar. Lastly, when they click the share button they are prompted to another screen where they can enter another user's email to share their calendar.

## Test Results

## Test Case 1:

| Figure 1.1 | Figure 1.2 |
| --- | --- |
|  |  |
| Figure 1.3 | Figure 1.4 |
|  |  |

* Test Case 1 deals with allowing the user to step up their class times in the schedules, including the starting and ending times for each class they have in a week. As the user navigates the app after logging in, they click the three dots on the top left of the screen which opens the bar on the left as seen in Figure 1.1. Then the user efficiently clicks ‘Create New’ and is prompted to a screen to fill out the information regarding their class as seen in Figure 1.2. The prompted screen in Figure 1.2 asks the user to fill out the following information: Class Name, Location, Start Time, End Time, Repetition, and Reminder Time. Once the user selects and types in the corresponding information (Figure 1.3), they press the ‘Add’ button which automatically adds this information to the calendar. On the Dashboard, the user can then view the calendar and click on the day to see the added information about the class along with the Class Name, Location, Start Time, and End Time. BuddyZen will also remind the user about this class 30 minutes before the Start Time of this class. The final calendar displayed in Figure 1.4 showcases the final result of the aforementioned process of adding a use to add class times to their planner.

Test Case 2:

| Figure 2.1 | Figure 2.2 |
| --- | --- |
|  |  |
| Figure 2.3 | Figure 2.4 |
|  |  |

* Test Case 2 displays how the user can add assignments to their planner. The user navigates similarly by logging into the app and then clicking the 3 black dots on the top left corner and is faced with the bar as shown in Figure 1.1. The user then presses ‘Tasks’ which pops up in Figure 2.1 and this allows the user to choose between adding a new task, deleting a task, or modifying a previous task. The user in this case wants to add a task so they select the ‘Add Task’ button and are prompted to the next window in Figure 2.2. In this layout, the user can choose between adding a specific type of task such as class, assignment, or other which can be their activity or clubs. The user in this case selects the ‘Assignment’ button which pops up the following screen in Figure 2.3. The user is prompted to add information regarding the assignment such as the Class Name, Upload Files, Due Date, Time, Priority, and Reminder. The user then fills in the information accordingly and presses the top right button in the shape of the pencil to finalize the assignment details. Once this is done, this information is automatically added to the planner and can be seen on the dashboard. It also shows the user the final view of the assignment information as seen in Figure 2.4 which includes all the details. Since the user chose the option ‘Y’ for priority, the color for the due date is Red and the app will also remind the user one day before the assignment is due as they choose. The user is now able to add their assignments in a straightforward and efficient manner.

Test Case 3:

| Figure 3.1 | Figure 3.2 |
| --- | --- |
|  |  |
| Figure 3.3 | Figure 3.4 |
|  |  |

* Test Case 3 deals with implementing the feature of being able to share schedules with other users who are using the application. The user starts by navigating to the taskbar again by clicking the 3 dots on the top left and clicking the share button at the bottom as shown in Figure 3.1. The user is then prompted to enter the email they would like to share their planner with as seen in Figure 3.2. Figure 3.3 showcases the user entering the email of the other user they prefer to share their information with and proceeding to click on the confirm button to finalize their decision. This allows both the users to now see each other’s schedule and one user’s perspective can be seen in Figure 3.4. Their schedule and the other person’s schedule can be seen in 2 different boxes with the option to scroll through either their or the other person’s tasks/events for the day. This Test Case successfully illustrates how the user can navigate BuddyZen to share their calendars with each other.

3. Code and UI Screen Design

3.1 Android Studio Implementation

3.1.1 Class User

import android.os.AsyncTask;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import android.content.Context;

import android.content.Intent;

import android.os.AsyncTask;

public class User {

private String userId;

private String username;

private String password;

private List<Event> events;

private Settings settings;

public User(String userId, String username, String password) {

this.userId = userId;

this.username = username;

this.password = password;

this.events = new ArrayList<>();

this.settings = new Settings();

}

public String getUserId() {

return userId;

}

public void setUserId(String userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public List<Event> getEvents() {

return events;

}

public void setEvents(List<Event> events) {

this.events = events;

}

public Settings getSettings() {

return settings;

}

public void setSettings(Settings settings) {

this.settings = settings;

}

static class UserLoginTask extends AsyncTask<String, Void, User> {

@Override

protected User doInBackground(String... credentials) {

try {

Connection connection = DriverManager.getConnection("jdbc:sqlserver://npcs.database.windows.net:1433;database=BuddyZen;user=test@npcs;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30");

String query = "SELECT \* FROM User WHERE username = {usernamefield.getText()} AND password = {passwordfield.getText()}";

try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, credentials[0]);

preparedStatement.setString(2, credentials[1]);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return new User(

resultSet.getString("userId"),

resultSet.getString("username"),

resultSet.getString("password")

);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

public class LoginTask extends AsyncTask<String, Void, User> {

private Context context;

public LoginTask(Context context) {

this.context = context;

}

@Override

protected User doInBackground(String... params) {

// Perform the background login task and return the User object

String enteredUsername = params[0];

String enteredPassword = params[1];

User user = loginUser(enteredUsername, enteredPassword);

return user;

}

@Override

protected void onPostExecute(User user) {

// Handle UI updates after login

if (user != null) {

// Login successful, navigate to the main dashboard

Intent intent = new Intent(context, MainActivity.class);

context.startActivity(intent);

} else {

// Handle login failure if needed

}

}

private User loginUser(String enteredUsername, String enteredPassword) {

return null;

}

}

3.1.2 Class Event

class Event {

private String eventId;

private String name;

private String time;

private String location;

private String instructor;

private String priority;

private String notificationSetting;

private String description;

private List<String> fileAttachments;

private User user;

private Calendar calendar;

Event(String eventId, String name, String time, String location, String instructor,

String priority, String notificationSetting, String description,

List<String> fileAttachments, User user, Calendar calendar) {

this.eventId = eventId;

this.name = name;

this.time = time;

this.location = location;

this.instructor = instructor;

this.priority = priority;

this.notificationSetting = notificationSetting;

this.description = description;

this.fileAttachments = fileAttachments;

this.user = user;

this.calendar = calendar;

}

public String getEventId() {

return eventId;

}

public void setEventId(String eventId) {

this.eventId = eventId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getTime() {

return time;

}

public void setTime(String time) {

this.time = time;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getInstructor() {

return instructor;

}

public void setInstructor(String instructor) {

this.instructor = instructor;

}

public String getPriority() {

return priority;

}

public void setPriority(String priority) {

this.priority = priority;

}

public String getNotificationSetting() {

return notificationSetting;

}

public void setNotificationSetting(String notificationSetting) {

this.notificationSetting = notificationSetting;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public List<String> getFileAttachments() {

return fileAttachments;

}

public void setFileAttachments(List<String> fileAttachments) {

this.fileAttachments = fileAttachments;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Calendar getCalendar() {

return calendar;

}

public void setCalendar(Calendar calendar) {

this.calendar = calendar;

}

static class CreateEventTask extends AsyncTask<Event, Void, Event> {

@Override

protected Event doInBackground(Event... events) {

try {

Connection connection = DriverManager.getConnection(“jdbc:sqlserver://npcs.database.windows.net:1433;database=BuddyZen;user=test@npcs;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30");

String query = "INSERT INTO Event VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, events[0].eventId);

preparedStatement.setString(2, events[0].name);

preparedStatement.setString(3, events[0].time);

preparedStatement.setString(4, events[0].location);

preparedStatement.setString(5, events[0].instructor);

preparedStatement.setString(6, events[0].priority);

preparedStatement.setString(7, events[0].notificationSetting);

preparedStatement.setString(8, events[0].description);

preparedStatement.executeUpdate();

}

return events[0];

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

@Override

protected void onPostExecute(User user) {

// Handle UI updates after login

if (user != null) {

// Login successful, navigate to the main dashboard

Intent intent = new Intent(context, MainActivity.class);

context.startActivity(intent);

} else {

// Handle login failure if needed

}

}

private User loginUser(String enteredUsername, String enteredPassword) {

return null;

}

}

3.1.3 Class Calendar

class Calendar {

private String calendarId;

private List<Event> events;

private User user;

Calendar(String calendarId, List<Event> events, User user) {

this.calendarId = calendarId;

this.events = events;

this.user = user;

}

public String getCalendarId() {

return calendarId;

}

public void setCalendarId(String calendarId) {

this.calendarId = calendarId;

}

public List<Event> getEvents() {

return events;

}

public void setEvents(List<Event> events) {

this.events = events;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

static class ViewCalendarTask extends AsyncTask<String, Void, List<Event>> {

@Override

protected List<Event> doInBackground(String... params) {

try {

Connection connection = DriverManager.getConnection("jdbc:sqlserver://npcs.database.windows.net:1433;database=BuddyZen;user=test@npcs;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30");

String query = "SELECT \* FROM Calendar WHERE calendarId = {calenderfield.getText()}";

try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, params[0]);

ResultSet resultSet = preparedStatement.executeQuery();

List<Event> events = new ArrayList<>();

while (resultSet.next()) {

// Create Event objects based on retrieved data

Event event = new Event(

resultSet.getString("eventId"),

resultSet.getString("name"),

resultSet.getString("time"),

resultSet.getString("location"),

resultSet.getString("instructor"),

resultSet.getString("priority"),

resultSet.getString("notificationSetting"),

resultSet.getString("description"),

new ArrayList<>(),

);

events.add(event);

}

return events;

}

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

@Override

protected void onPostExecute(User user) {

// Handle UI updates after login

if (user != null) {

// Login successful, navigate to the main dashboard

Intent intent = new Intent(context, MainActivity.class);

context.startActivity(intent);

} else {

// Handle login failure if needed

}

}

private User loginUser(String enteredUsername, String enteredPassword) {

return null;

}

}

3.1.4 Class Settings

class Settings {

static class ChangePasswordTask extends AsyncTask<String, Void, Boolean> {

@Override

protected Boolean doInBackground(String... params) {

try {

Connection connection = DriverManager.getConnection("jdbc:sqlserver://npcs.database.windows.net:1433;database=BuddyZen;user=test@npcs;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30l");

String query = "UPDATE User SET password = {passwordfield.getText()} WHERE userId = {userfield.getText()}";

try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setString(1, params[0]);

int rowsAffected = preparedStatement.executeUpdate();

return rowsAffected > 0;

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

@Override

protected void onPostExecute(User user) {

// Handle UI updates after login

if (user != null) {

// Login successful, navigate to the main dashboard

Intent intent = new Intent(context, MainActivity.class);

context.startActivity(intent);

} else {

// Handle login failure if needed

}

}

private User loginUser(String enteredUsername, String enteredPassword) {

return null;

}

}

3.1.5 Class MainActivity

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

new User.UserLoginTask().execute("username", "password");

List<String> fileAttachments = new ArrayList<>();

Event newEvent = new Event(

"eventId",

"EventName",

"2023-11-09T12:00:00",

"EventLocation",

"InstructorName",

"High",

"15 minutes before",

"EventDescription",

fileAttachments,

);

new Event.CreateEventTask().execute(newEvent);

new Calendar.ViewCalendarTask().execute("calendarId");

new Settings.ChangePasswordTask().execute("newPassword");

}

}

3.2 Azure Database Setup

3.2.1 Azure Database Query

-- Create the BuddyZen database

CREATE DATABASE BuddyZen;

-- Use the BuddyZen database

USE BuddyZen;

-- Create the User table

CREATE TABLE User (

userId UNIQUEIDENTIFIER PRIMARY KEY,

username NVARCHAR(255) NOT NULL,

password NVARCHAR(255) NOT NULL,

settings NVARCHAR(MAX),

);

-- Create the Event table

CREATE TABLE Event (

eventId UNIQUEIDENTIFIER PRIMARY KEY,

name NVARCHAR(255) NOT NULL,

time DATETIME NOT NULL,

location NVARCHAR(255),

instructor NVARCHAR(255),

priority NVARCHAR(50),

notificationSetting NVARCHAR(50),

description NVARCHAR(MAX),

fileAttachments NVARCHAR(MAX),

userId UNIQUEIDENTIFIER FOREIGN KEY REFERENCES User(userId),

calendarId UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Calendar(calendarId),

);

-- Create the Calendar table

CREATE TABLE Calendar (

calendarId UNIQUEIDENTIFIER PRIMARY KEY,

userId UNIQUEIDENTIFIER FOREIGN KEY REFERENCES User(userId),

);

-- Create an index on the userId column in the Event table for faster queries

CREATE INDEX IX\_Event\_UserId ON Event(userId);

-- Create an index on the userId column in the Calendar table for faster queries

CREATE INDEX IX\_Calendar\_UserId ON Calendar(userId);

3.2.2 Java Code For Interaction with JDBC

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class DatabaseExample {

public static void main(String[] args) {

String jdbcUrl = "jdbc:sqlserver://npcs.database.windows.net:1433;database=BuddyZen;user=test@ipro497;password={your\_password\_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=\*.database.windows.net;loginTimeout=30";

try (Connection connection = DriverManager.getConnection(jdbcUrl)) {

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM User");

} catch (Exception e) {

e.printStackTrace();

}

}

}

3.3 UI Design

|  | Logo of BuddyZen   * Intuitive and clean design * Attractive to users |
| --- | --- |
|  | Color Scheme of BuddyZen   * Blue shades used so that the User Experience is more relaxed |
|  | Login Page of BuddyZen   * Prompts the user to enter their login credentials. * If the user forgot their password or has yet to create an account, the user can select the Create button. * Text fields are provided in the UI so that the user may enter their details and log in. * If a user enters an invalid input, an error message is displayed to the user to re-enter their information |
|  | Central Calendar Page of BuddyZen   * Main Dashboard * Presents an organized overview of the user’s schedule. * The central area features a monthly calendar view. * Consists of motivational quotes and refreshing graphics, enhancing the user experience. |
|  | Scrollable event popup of BuddyZen   * Appears after clicking on the date. * Contains a list of all past and upcoming schedules. * Schedules that are past appear in red, immediately upcoming schedules show in green, and later schedules are all white. * Arrow buttons are provided on either side of the popup to help navigate between different dates quickly without needing to press the date itself. |
|  | Sidebar View of BuddyZen   * Sidebar view is displayed after pressing the three dots at the upper left of the main dashboard. * Consists of tabs for Calendar, Tasks, Activities, and Settings * Provides quick access to priority tasks and upcoming deadlines * Users can add new tasks with a clear “Create New” button * Users can share the calendar with another user on the app by selecting the “Share” button. |
|  | Share View of BuddyZen   * Allows user to enter email of the other BuddyZen user to share their calendar |
|  | Example of entered email of BuddyZen |
|  | Shared View of Two Results of BuddyZen |
|  | Tasks button from Sidebar of BuddyZen |
|  | Add Tasks View of BuddyZen |
|  | Add Assignment View of BuddyZen |
|  | Example of Assignment in BuddyZen   * Each task can be categorized by type (assignment, personal, club-related). * Priority levels are easily adjustable by selecting the pencil symbol to edit. * Checkbox allows users to mark tasks as complete. * The interface also supports drag-and-drop functionality for effortless task rearrangement. * Assignment events display due dates and associated courses, while extracurricular events provide information on the club, location, and time. |
|  | Create New Class View in BuddyZen |
|  | Filling in the fields in Create New Class in BuddyZen |
|  | View Newly Created Class in BuddyZen |
|  | Class View of BuddyZen   * Clicking on a calendar event opens a detailed view. * For classes, it shows the course name, location, and professor details. * Users can quickly edit or delete events directly from this view, enhancing the overall user experience. |
|  | Settings View of BuddyZen   * Settings view consists of tabs for Account, Notifications, Appearance, Privacy and Security, Help Desk and About. * User can find their profile information and customization options. * Settings view allows users to set preferences for notification timings, choose app themes, and manage account details. * The UI ensures a straightforward and visually appealing layout. |