System Design Document

IWAC Conference Application for Heather Falconer

PenUltimate

Brett Palmer, Monica Agneta, George Pitt, Rebecca Sonnemann, Ben Caras

3 November 2025

Table of Contents

1 Introduction

This capstone project is being completed in partial fulfillment of the requirements for the B.S. in Computer Science degree for the University of Maine. The client for this project is part of the Association for Writing Across the Curriculum (AWAC), which organizes the International Writing Across the Curriculum Conference (IWAC), a multi-day event that hosts both educators and researchers. In 2027, UMaine is hosting this conference. The client is interested in having a mobile conference application that could be repurposed for future conferences. The client noted having issues with last year's application, such as having to scroll through too many events and being unable to filter through them (Whova, 2025). The client also mentioned that when a previous app was used on mobile devices, it opened up a web page that was not optimal (LineUpr, 2025). The problem that the client is trying to solve is finding an effective way to allow in-person attendees to navigate the session schedule and find the info they need, as well as creating a way for virtual attendees to have more of a conference experience.

1.1 Purpose of This Document

The purpose of this System Design Document (SDD) is to detail the architectural and design specifications for the IWAC Conference Application being developed by the PenUltimate team. This document translates the functional and non-functional requirements defined in the System Requirements Specification (SRS) document into a framework that guides implementation of the product. The System Design Document is intended for the PenUltimate development team and organizers of the IWAC conference to ensure a shared understanding of the system's architecture. This document begins with an overview of the system architecture, followed by detailed descriptions of each function and the data flow. It then goes on to describe the user database and files intended to be used by the system. The document concludes with a table that depicts which system components satisfy which functional requirements from the SRS.

1.2 References

Association for Writing Across the Curriculum. (2025). *IWAC 2025*. https://iwac2025.lineupr.com/iwac-2025/

dbdiagram.io. (2025). Dbdiagram.io: Database relationship diagrams design tool. Holistics Software. https://dbdiagram.io/d

draw.io. (2025). *Draw.io: Free flowchart maker and diagrams online.* draw.io Ltd. https://app.diagrams.net/

Fowler, M. (1997). *UML Distilled: A brief guide to the standard object modeling language*. Addison-Wesley. http://ci.nii.ac.jp/ncid/BA65029497

LineUpr. (2025). *The Event application solution to boost your event communication – LineUpr*. LineUpr GmbH. https://lineupr.com/en

Whova. (2025). *Whova: Award-winning Event Apps and Event Management*. Whova, Inc. https://whova.com/

2 System Architecture

This section outlines the overall design of the IWAC Conference application, detailing the overall architecture and how the components interact to fulfill the functional requirements defined in the System Requirements Specification. The architecture is first showcased by a layered architecture diagram created with draw.io (2025) that gives a high-level view of the data flow through the application. Complementing this, a UML class diagram created with dbdiagram.io (2025) shows the decomposition of the system into the major components, specifying variables, methods, and relationships. Together, these models illustrate both the system's logical and structural design and the user interaction with it.

2.1 Architectural Design

Figure 2.1 is a Layered architecture diagram, this diagram has three different layers. The top layer is the user interface layer which contains all pages that the users and systems administrator are able to view. The application layer contains all of the different actions they are able to perform. These actions change the data within the database that is noted on the data layer of the diagram.

The IWAC application will be available on both Android and Apple operating systems, specifically on Android 12+ and Apple iOS 16+. The application will serve as a virtual environment where they will be able to interact with conference events as well as conference attendees and presenters. Allowing users to view events and create their own personal itinerary of the events which they select. They will be able to message other users as well as comment on presentations and allow for presenters to answer their questions virtually. The information related to messaging, commenting, and creation of personal calendars as well as other key functions will be stored within our application's database. The application will also be connected to other external applications such as Google Drive which will store all presentation materials where users can be redirected to view this information, YouTube where users will be able to watch prerecorded presentations or livestreams of the presentations, and Google Maps which will help conference attendees gain a better understanding of where the events and presentations will take place.

The application will primarily be written in TypeScript using the Expo full-stack React Native framework. The system will be tested using Jest which is a common testing framework for React Native. The application will use Supabase for a PostgreSQL database, which is an open source database software. Our client will enter the information for each conference via Supabase's user-friendly interface. Our team believes that TypeScript with the Expo full-stack React Native framework will work best for our application because of the fact that the application needs to be usable on both Android and iOS operating systems. The Jest framework for testing is the most understood amongst our team, and the database will allow our client to be able to input data into the application easily given Supabases user-friendly UI and easy data entry.

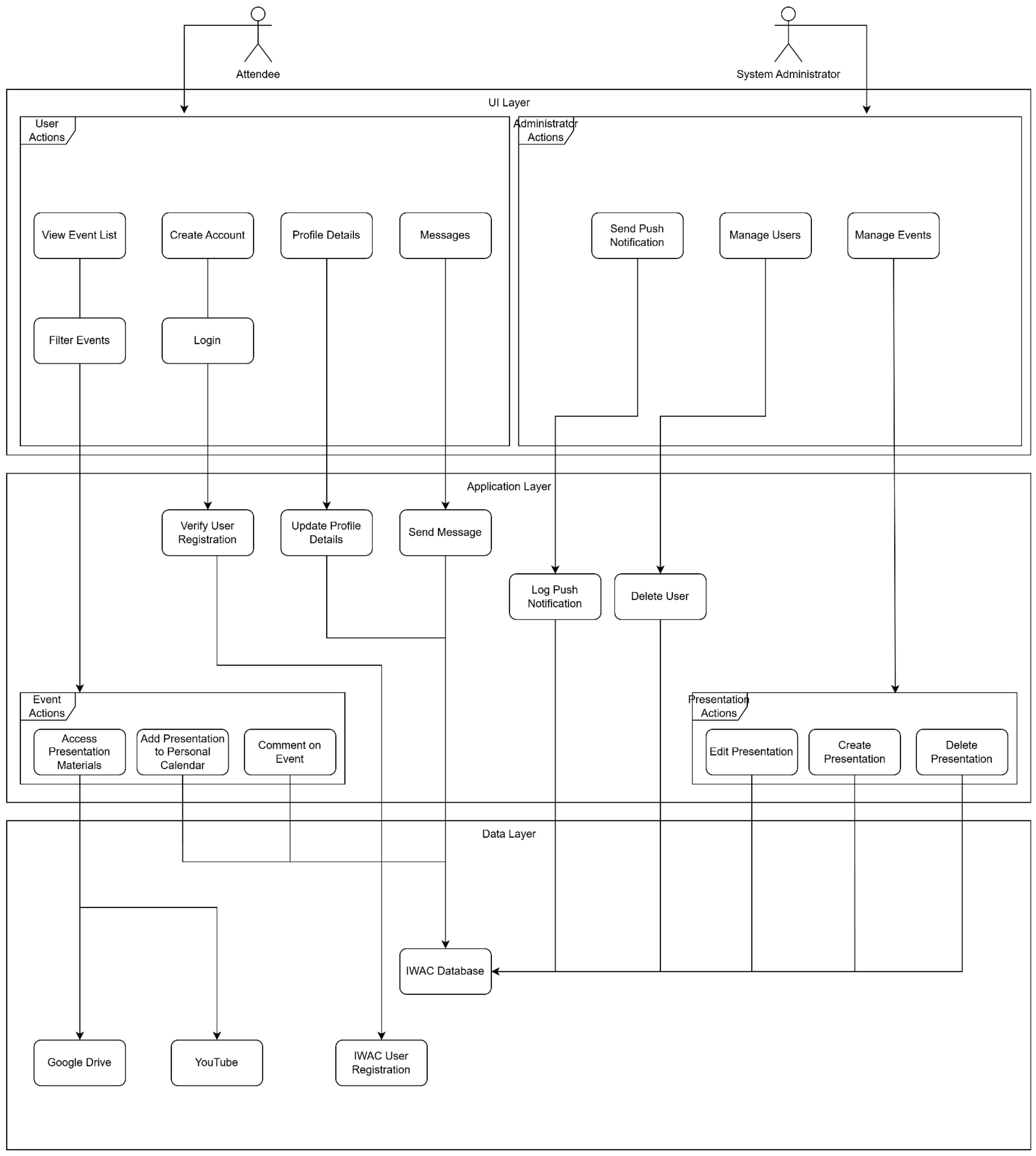

Figure 2.1 - Layered Architecture Diagram

2.2 Decomposition Description

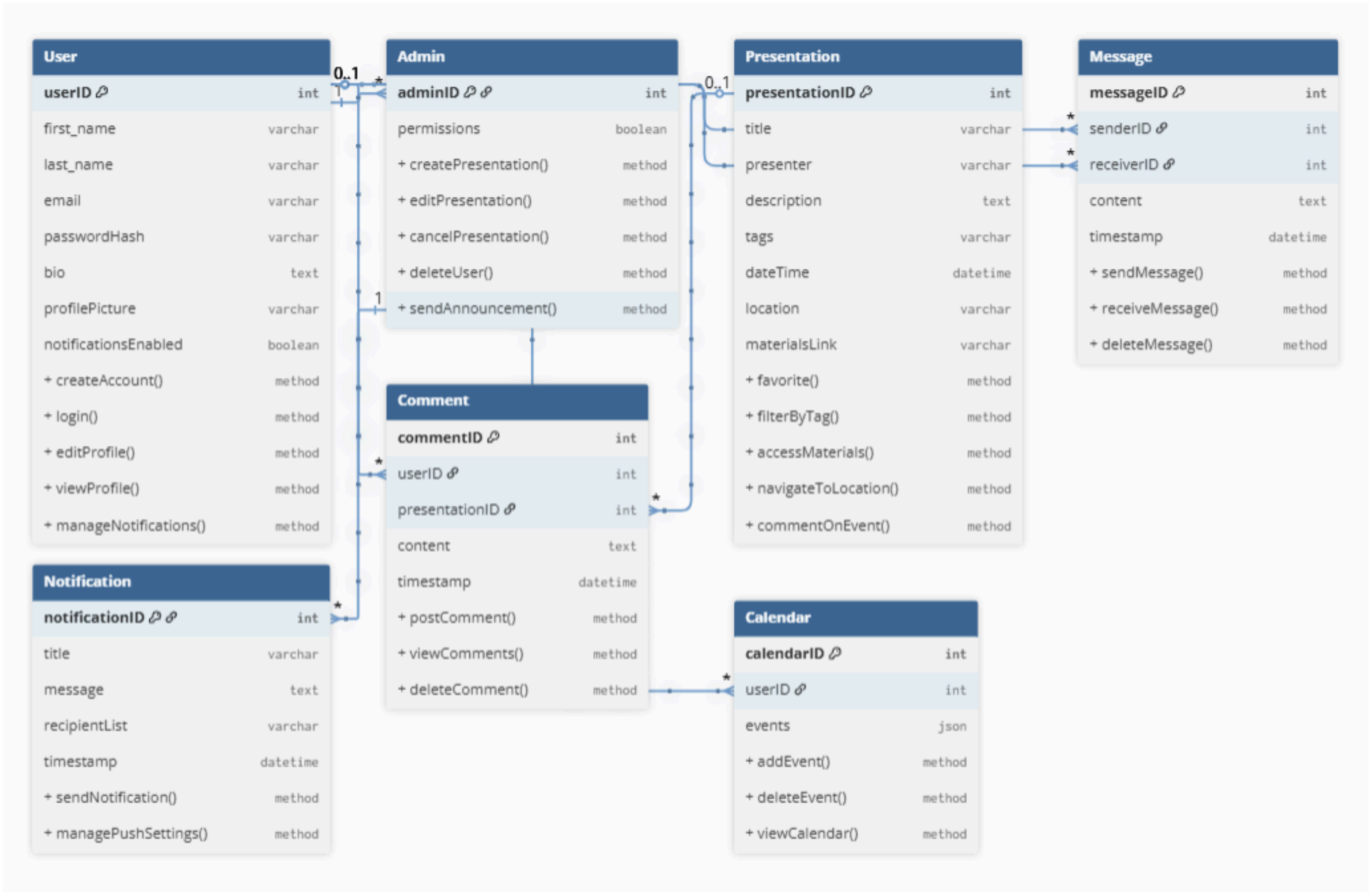

Figure 2.2 - IWAC Conference App Class Diagram

Figure 2.2 is a class diagram illustrating the various objects in our application, including, but not limited to, user, admin, and calendar. Some of these objects extend from other objects. For example, an Admin is a type of User with additional permissions, meaning that every element of a User, such as first_name, last_name, and email. is also an element of Admins. The encapsulation created by having these different objects, and the inheritance they gain, such as with the above example, is why our application is object-oriented, and why we used class diagrams to illustrate the implementation design of our system. On the left side of each object’s “box”, you will find the data that is held within that object. To the right, you can see what type of data it is, including but not limited to text, an integer, a floating-point number, or a boolean. For example, we can see that the Comment object has a commentID to identify that specific comment, a UserID to identify who made that comment, and a presentationID to know what presentation it was commented under, all saved as integers. The comment object also holds the text contents of the message, and the timestamp of when it was posted. Lastly, users can also view comments, as well as post and delete their own, all saved as different methods.

A model view controller design was used for this, as it allows us to separate the application’s logic into distinct components. This makes the system easier to maintain and update in the future, and allows users to view data and swap between different views, or screens. Using the previous example of the Comment object, the commentID, userID, and presentationIDs would all be saved to our database and act as our model, while the methods addEvent(), viewCalendar(), and viewEvent() would act as the view. In general, objects get implemented into our system by having their attributes saved to the database, while the methods of that object are saved in the application layer to be used as the model.

3 Persistent Data Design

The persistent data design of the IWAC system defines how information is stored, organized, and accessed across the application. This design ensures consistency and scalability for managing events and user interactions. It includes the entity relationships and file structures that collectively create the system's functionality. The following sections describe the database descriptions and associated files used to maintain data persistence within the IWAC Application.

3.1 Database Descriptions

Figure 3.1 presents the entity-relationship diagram for the IWAC database used by the IWAC Application. It provides a high-level overview of the system's data structure and inner table relationships. Each table in the diagram represents an entity of our conference management system, such as users, conferences, and events. The lines between the tables indicate the relationships between the entities, such as one-to-one, one-to-many, and many-to-many. Within each table, a key icon designates the primary key that uniquely identifies each record of the table. Also, the link icon marks foreign keys that are used to link related tables. Together, these visual cues clarify how data flows between components of the system.

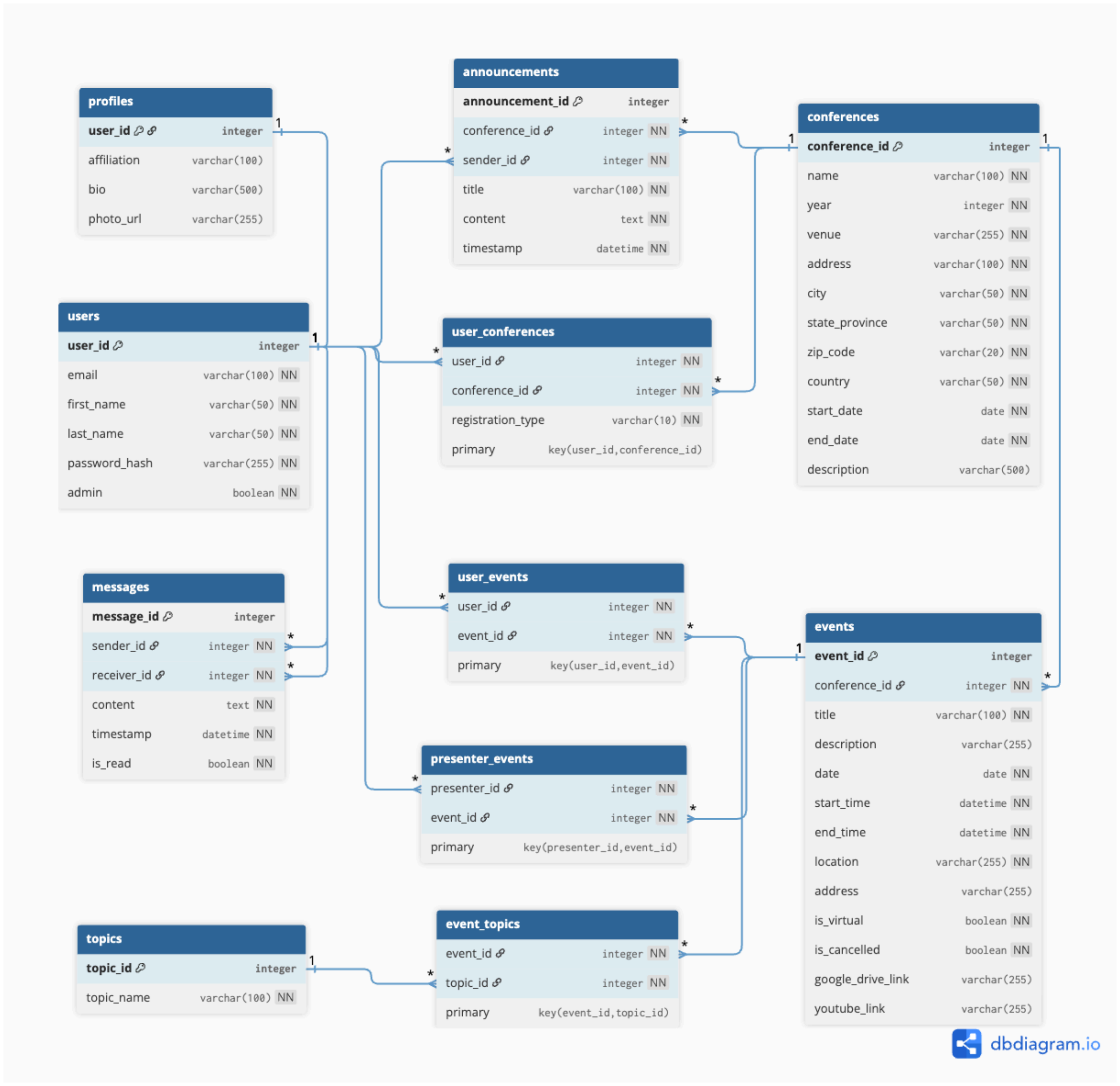

Figure 3.1 Entity-Relationship Diagram for the Conference Management Database

The following tables (Table 3.1.1 through Table 3.1.10) provide a detailed breakdown of each entity from Figure 3.1, outlining the structure and attributes of the IWAC database. Each field is presented with its record type, size, description, and whether it is required. Primary keys are shown in bold, while foreign keys are shown in italics. Together, these tables describe how each database component stores and organizes information that is essential to manage the IWAC conferences.

Table 3.1.1 Users Table Definition

Field	Type	Size	Description	Required
user_id	INT	-	Auto-incremented unique identifier for each user.	Yes
email	VARCHAR	100	User's conference-registered email. Must be unique.	Yes
first_name	VARCHAR	50	User's first name.	Yes
last_name	VARCHAR	50	User's last name.	Yes
password_hash	VARCHAR	255	Hashed and salted password for login.	Yes
admin	BOOLEAN	-	True if the user is an admin, false otherwise.	Yes

Table 3.1.2 Profiles Table Definition

Field	Type	Size	Description	Required
user_id	INT	-	References Users(user_id). Unique identifier for each user.	Yes
affiliation	VARCHAR	100	User's institution or organizational affiliation.	No
bio	VARCHAR	500	Short biography for the user to describe themselves.	No
photo_url	VARCHAR	255	Link to the user's optional profile picture.	No

Table 3.1.3 Conferences Table Definition

Field	Type	Size	Description	Required
conference_id	INT	-	Auto-incremented unique identifier for each conference.	Yes
name	VARCHAR	100	Official conference title.	Yes
year	INT	-	Year of the conference.	Yes

venue	VARCHAR	255	Name of the venue where the conference is held.	Yes
address	VARCHAR	100	Street address of the venue.	Yes
city	VARCHAR	50	City of the venue.	Yes
state_province	VARCHAR	50	State or province of the venue.	Yes
zip_code	VARCHAR	20	ZIP code of the venue.	Yes
country	VARCHAR	50	Country name of the venue	Yes
start_date	DATE	-	Start date of the conference	Yes
end_date	DATE	-	End date of the conference	Yes
description	VARCHAR	500	Brief overview of the conference and its theme	No

Table 3.1.4 UserConferences Table Definition

Field	Type	Size	Description	Required
user_id	INT	-	References Users(user_id). Unique identifier for each user.	Yes
conference_id	INT	-	References Conferences(conference_id). Unique identifier for each conference.	Yes
registration_type	ENUM (‘in_person’, ‘virtual’)	10		Yes

Table 3.1.4 Events Table Definition

Field	Type	Size	Description	Required
event_id	INT	-	Auto-incremented unique identifier for each event.	Yes
conference_id	INT	-	References Conferences(conference_id). Unique identifier for each conference.	Yes

title	VARCHAR	100	Title of the event.	Yes
description	VARCHAR	255	Brief overview of the event and its theme.	No
date	DATE	-	Scheduled date of event.	Yes
start_time	DATETIME	-	Start time of the conference.	Yes
end_time	DATETIME	-	End time of the conference.	Yes
location	VARCHAR	255	Building name and room name (e.g., “Neville Hall, Room 116”). Is “Virtual” if held entirely online.	Yes
address	VARCHAR	255	Street address of the event, if in-person. The event’s other location attributes can be assumed to be the same as the conference.	No
is_virtual	BOOLEAN	-	True if users can participate in the event virtually, false by default.	Yes
is_cancelled	BOOLEAN	-	True if the event gets cancelled, false by default.	Yes
google_drive_link	VARCHAR	255	Link to slides and/or materials.	No
youtube_link	VARCHAR	255	Link to livestream or video.	No

Table 3.1.5 UserEvents Table Definition

Field	Type	Size	Description	Required
user_id	INT	-	References Users(user_id). Unique identifier for each user.	Yes
event_id	INT	-	References Events(event_id). Unique identifier for each event.	Yes

Table 3.1.6 PresenterEvents Table Definition

Field	Type	Size	Description	Required
presenter_id	INT	-	References Users(user_id). Unique identifier for each user, in this case a presenter.	Yes

event_id	INT	-	References Events(event_id). Unique identifier for each event.	Yes

Table 3.1.7 EventTopics Table Definition

Field	Type	Size	Description	Required
event_id	INT	-	References Events(event_id). Unique identifier for each event.	Yes
topic_id	INT	-	References Topics(topic_id). Unique identifier for each topic.	Yes

Table 3.1.8 Messages Table Definition

Field	Type	Size	Description	Required
message_id	INT	-	Auto-incremented unique identifier for each message.	Yes
sender_id	INT	-	References users(user_id). The user who sent the message.	Yes
reciever_id	INT	-	References users(user_id). The user who receives the message.	Yes
content	VARCHAR	1000	Text content of the message.	Yes
timestamp	DATETIME	-	Date and time when the message was sent.	Yes
is_read	BOOLEAN	-	True if the receiver has opened the message; otherwise false.	Yes

Table 3.1.9 Topics Table Definition

Field	Type	Size	Description	Required
topic_id	INT	-	Auto-incremented unique identifier for each topic.	Yes

topic_name	VARCHAR	100	Label or subject area describing the description of an event.	Yes

3.1.10 Announcements Table Definition

Field	Type	Size	Description	Required
announcement_id	INT	-	Auto-incremented unique identifier for each announcement or push notification.	Yes
conference_id	INT	-	References Conferences(conference_id). Identifies which conference this announcement belongs to.	Yes
title	VARCHAR	150	Short title or headline of the announcement.	Yes
content	VARCHAR	1000	Main body text of the announcement message.	Yes
timestamp	DATETIME	-	Date and time when the announcement was created or sent.	Yes
sender_id	INT	-	References Users(user_id). Identifies the user (an admin) who posted the announcement.	Yes

3.2 File Descriptions

The following tables (Table 3.2.1 through Table 3.2.2) describe the files used in the IWAC system. Each file contains its name, data type, size, and a description of what it represents. These tables describe each file that is necessary within the IWAC conference application.

Table 3.2.1 Banner Files Table Definition

Field	Type	Size	Description	Required
file_name	VARCHAR	255	Actual file name stored in the server	Yes

			directory.	
file_path	VARCHAR	500	Full relative or absolute path to the banner file.	Yes
conference_id	INT	-	References Conferences(conference_id). Identifies which conference this banner belongs to.	Yes
upload_date	DATETIME	-	Date and time the banner was uploaded.	Yes
uploaded_by	INT	-	References Users(user_id). Identifies the admin who uploaded the banner.	Yes
file_type	VARCHAR	10	MIME type of the file.	Yes
file_size	INT	20	Size of the file in bytes.	Yes
resolution	INT	200	Image dimensions.	Yes

Table 3.2.2 Profile Picture Files Table Definition

Field	Type	Size	Description	Required
file_name	VARCHAR	255	File name of the uploaded picture.	Yes
file_path	VARCHAR	500	Full relative or absolute path to the image.	Yes
user_id	INT	-	References Users(user_id). The user who owns the profile picture.	Yes
upload_date	DATETIME	-	Date and time the picture was uploaded.	Yes
file_type	VARCHAR	10	MIME type of the file.	Yes
file_size	INT	-	Size of the file in bytes.	Yes
resolution	VARCHAR	20	Image dimensions.	Yes

4 Requirements Matrix

Table 4 is a Requirements Matrix Table that shows each of our functional requirements listed on the left, previously outlined in our SRS. The middle column holds the corresponding methods that address each of these issues, while the right column holds the class that uses the corresponding method in that row. Having each Functional Requirement be dealt with by at least one method ensures that we complete all of our Functional Requirements when developing this application. These methods, and which objects they are associated with, as well as additional methods not listed on this table, are laid out in figure 2.2.

Table 4 Requirements Matrix Table

FR Number and Name	**Corresponding Method(s)**	**Class(es) in Which the Method is Used.**
1- Create Account	createAccount()	User
2- Login to Account	login()	User
3- Edit Profile Details	editProfile()	User
4- View Other User Profiles	viewProfile()	User
5- Message User	sendMessage()	Message
6- Manage Push Notifications	managePushSettings()	Notification
7- Filter presentations	filterByTag()	Presentation
8- Access Presentation Materials	accessMaterials()	Presentation
9- Add Presentation to Personal Calendar	favorite()	Presentation
10- Access Personal Calendar	viewCalendar()	Calendar
11- Navigate to Event Location	navigateToLocation()	Presentation

12- Comment on Event	commentOnEvent()	Presentation
13- Access Previous Years' Presentations	filterByTag()	Presentation
14- Create Presentation	createPresentation()	Admin
15- Edit Presentation	editPresentation()	Admin
16- Delete Presentation	cancelPresentation()	Admin
17- Delete User	deleteUser()	Admin
18- Send Push Notifications	sendAnnouncement() / sendNotification()	Admin, Notification

Appendix A – Agreement Between Customer and Contractor

The PenUltimate team is responsible for developing the application as outlined within this document. The development includes application features, user interface requirements, user and administrator actions, data storage, and database management. The development can be done using open source software to base our work off of, and future developers will have access to our code and the database to be able to update it in the future as needed, as our involvement with building this application will end after May 2026.

Should either party wish to make changes to this document, both parties must meet ahead of time and mutually agree on said change. If Heather Falconer is the party requesting these changes, they should reach out to our client liaison, Monica Agneta, with their requests. Similarly, should we request any changes to this document, Monica will reach out to Heather with our requests.

Signature: Brett Palmer Date: 11/2/2025

Signature: Monica Agneta Date: 11/2/2025

Signature: [illegible] Date: 11/2/2025

Signature: [illegible] Date: 11/2/2025

Signature: George Pitt Date: 11/2/2025

Signature: ______________________________ Date: 11/2/2025

Appendix B – Team Review Sign-off

All team members have signed to acknowledge they have reviewed this document and agreed on both its content and format. If team members have minor disagreements, they may state them in the comments area.

Rebecca Sonnemann
Signature: [illegible] Date: 11/02/2025
Comments:

Brett Palmer
Signature: Brett Palmer Date: 11/02/2025
Comments:

Monica Agneta
Signature: Monica Agneta Date: 11/02/2025
Comments:

Ben Caras
Signature: [illegible] Date: 11/02/2025
Comments:

George Pitt
Signature: George Pitt Date: 11/02/2025
Comments:

Appendix C – Document Contributions

Brett

Brett designed the database structure and represented it in diagram 3.1, created tables 3.1.1 through 3.1.6, and wrote the text within section 3.1.

Brett estimates that he did 20% of the work for this document.

Monica

Monica wrote section 1, wrote the introduction for section 2, and created the decomposition description diagram, figure 2.2.

Monica estimates that she did 20% of the work for this document.

George

George wrote the diagram description for section 2.2, as well as section 4.

George estimates that he did 20% of the work for this document.

Rebecca

Rebecca wrote the introductions for section 3 and section 3.2, created tables 3.1.7 through 3.1.10, and did the tables for section 3.2.

Rebby estimates that she did 20% of the work for this document.

Ben

Ben wrote section 2.1 and the diagram associated with this section. Ben also wrote Appendix A.

Ben estimates that he did 20% of the work for this document.