System Design Document

IWAC Conference Application for Heather Falconer

PenUltimate

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

17 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 can 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 an interactive 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 SDD 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 SRS. 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

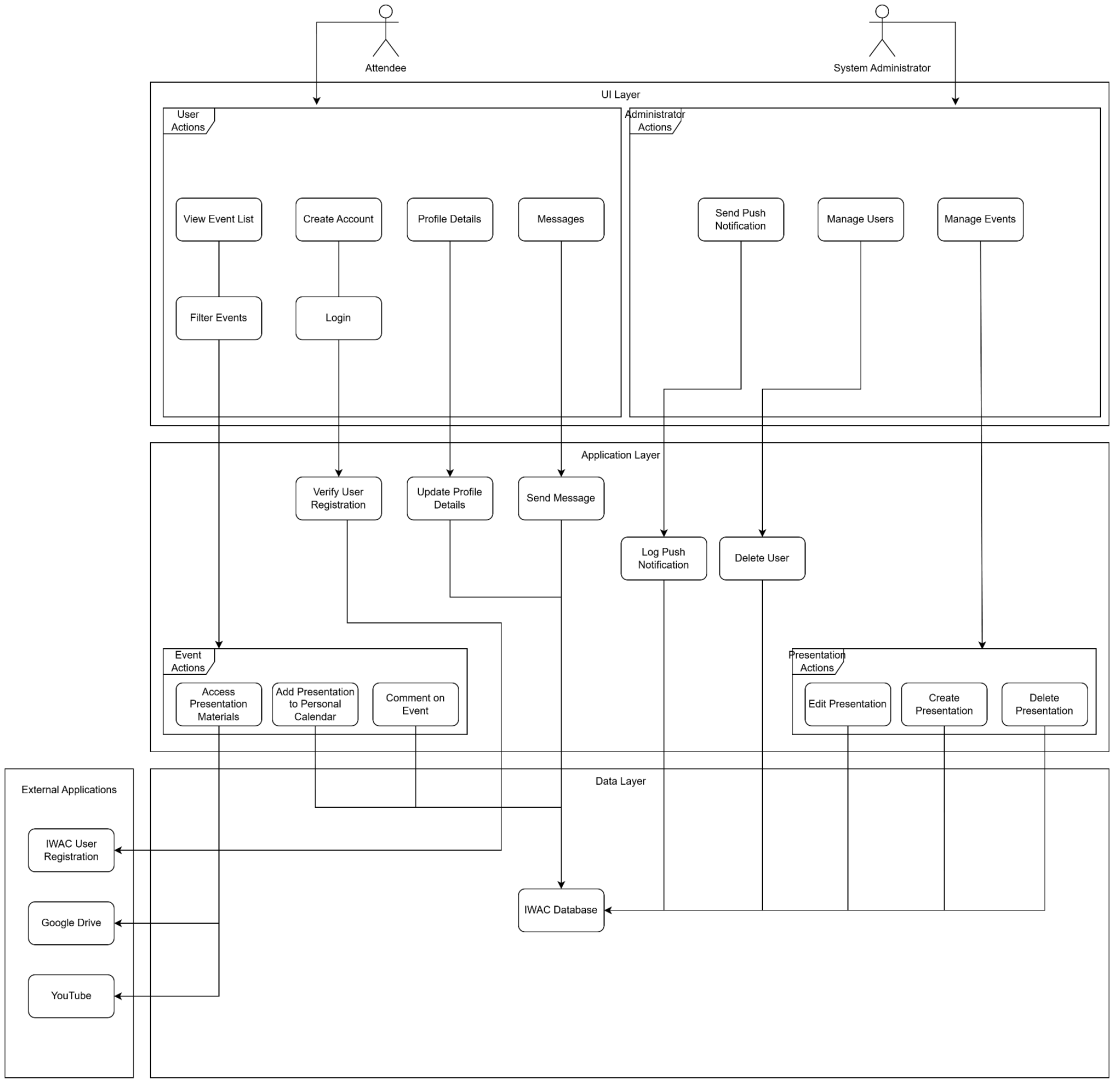


Figure 2.1.1 - Layered Architecture Diagram

Figure 2.1.1 is a layered architecture diagram with three distinct 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. Next to the data layer are external applications which are systems that the IWAC application will use but are not present within the IWAC application.

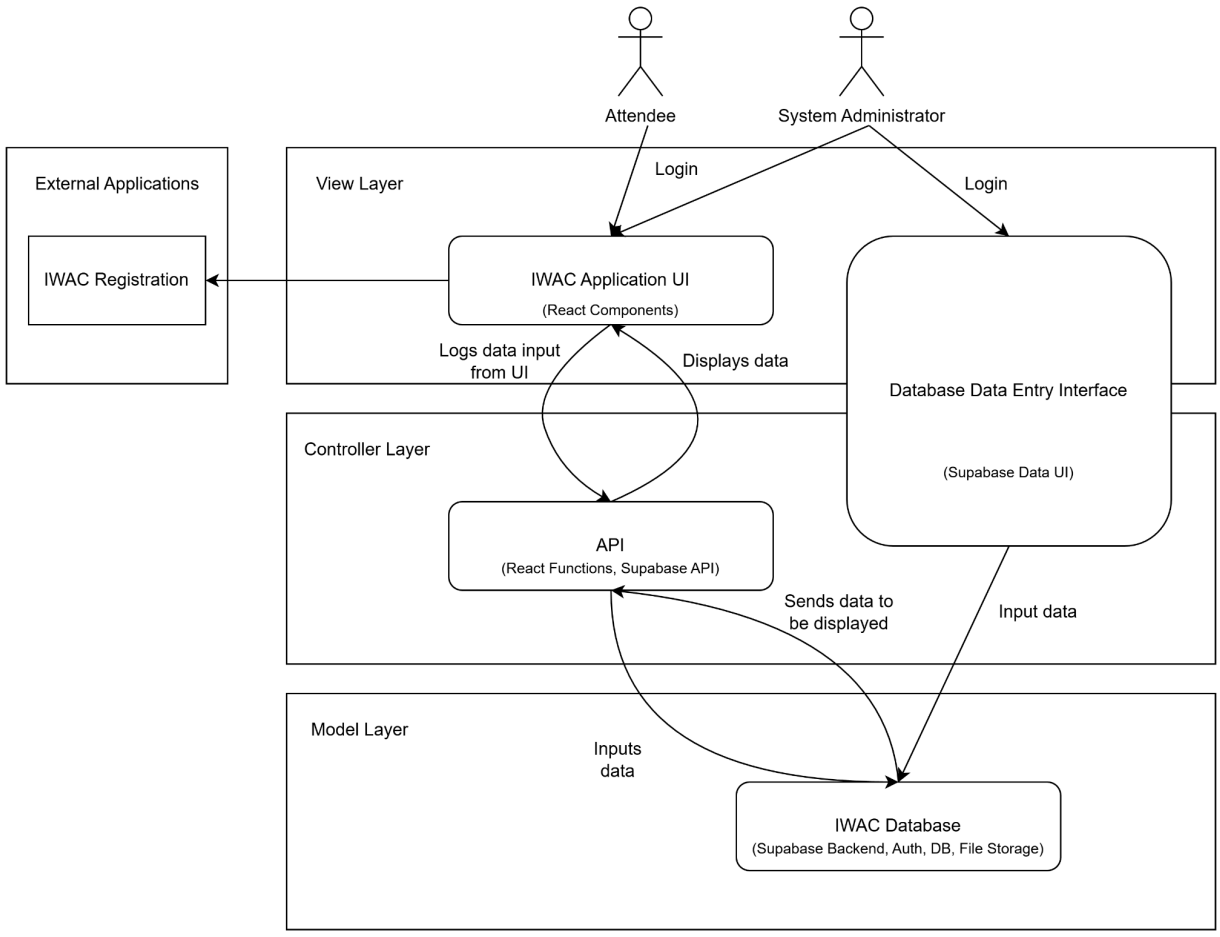


Figure 2.1.2 - MVC Logical Architecture Diagram

Figure 2.1.2 is a Model-View-Controller logical architecture diagram; it separates the system's components into three distinct layers. The View layer which contains the user interface of our application as well as the user interface used by the database which is how the client will input information into the database. The Controller layer which contains the API calls that will be made when information is entered into the user interface or when entered into the database. The Model layer contains the Supabase database that will be used by the IWAC application.

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 the attendees will be able to interact with conference events as well as conference attendees and presenters. The system will allow users to view a list of conference events and create their own personal itinerary. Users will be able to message other users, comment on presentations, and receive responses from the presenters. 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 linked 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, a common testing framework for React Native. The application will use Supabase for a PostgreSQL database, an open source database software. The client will enter the information for each conference via Supabase's 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 the fact that Supabases UI and data entry aligns with the clients expertise.

2.2 Decomposition Description

The IWAC Application is an object-oriented system. The User represents anyone who uses the app. The Admin represents a subclass of users that can do everything a User can, but with additional privileges such as managing presentations and other user accounts. The Notification object handles push notifications sent to the user's device. The Presentation object represents a given panel being held at the conference. The Comment object is associated with the Presentation object, allowing users to leave comments. The Message object handles private communication between users, enabling one user to send a direct message to another user. Lastly, the Calendar object stores each user's favorited presentations, creating unique personalized calendars for each attendee. Note that some of these objects extend from other objects, such as Admin and User. The interconnectedness of these objects, the ability to individually alter or manage any one of them without affecting others, and the inheritance some objects gain when extending from others, explain why our application is object-oriented. These motivations also justify our use of class diagrams to illustrate the implementation design of our system.

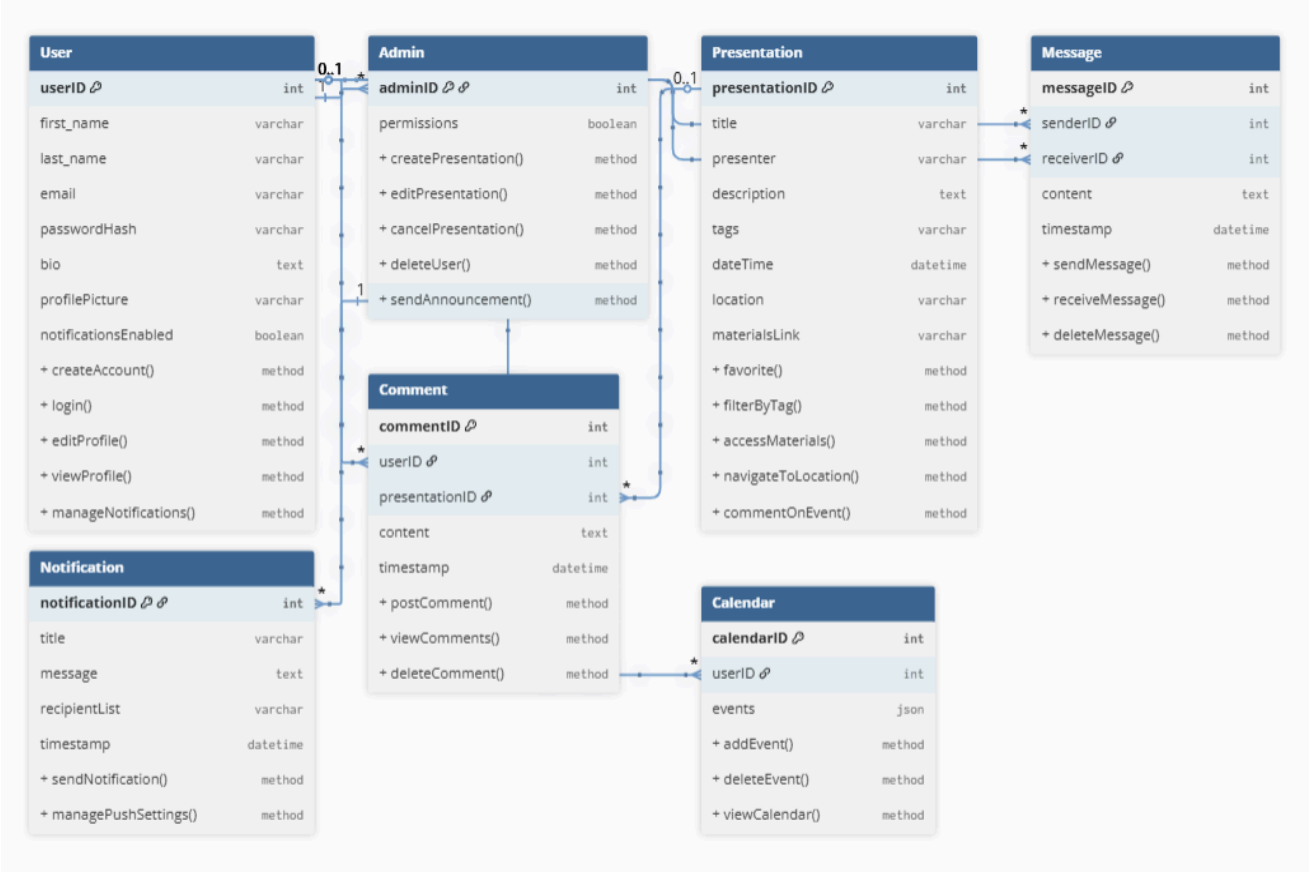


Figure 2.2 - IWAC Conference App Class Diagram

Figure 2.2 is a class diagram illustrating the various objects in our application. The labels “0”, “1”, and “*” on each side the lines connecting objects indicate how many instances of one object may be associated with another, with “*” being shorthand for “many.” For instance, many comments can be left under a single presentation, so the line between the objects shows a “*” on the comment side and a “1” on the presentation side. Furthermore, the diagram shows each object with its associated data and methods listed to the left, while the right side specifies the type of data held or whether it is a method. For example, the Comment object has a commentID to identify that specific comment, a userID to identify who made that comment, and a presentationID to indicate which presentation it was commented under, all stored as integers. It also contains the text contents of the message and the timestamp of when it was posted. Lastly, users can perform the actions of viewing, posting, and deleting comments, which are represented by the methods viewComment(), postComment(), and deleteComment(), respectively.

A Model View Controller design pattern was used for the system, as it allows us to separate the application’s logic into distinct components. This separation makes the system easier to maintain and update in the future, and allows users to view and modify data in a controlled manner as they interact with the application’s screens. First, the Model directly

manages our application's data and the rules governing that data, ensuring that all stored information is consistent. The View is the interface that users see and interact with in the app, such as screens containing search bars, buttons, input fields, and other visual components. The Controller links the two, taking data in the Model and displaying it in the View, or receiving input from the View and updating data in the Model. Using the previous example of the Comment object, the commentID, userID, and presentationID values would all be stored in our database, which is the Model. Methods such as addEvent(), viewCalendar(), and viewEvent() would belong to the Controller, as they retrieve and update the appropriate information based on events. The Events screen would serve as the View, since this is what users will interact with in order to comment under a presentation.

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.

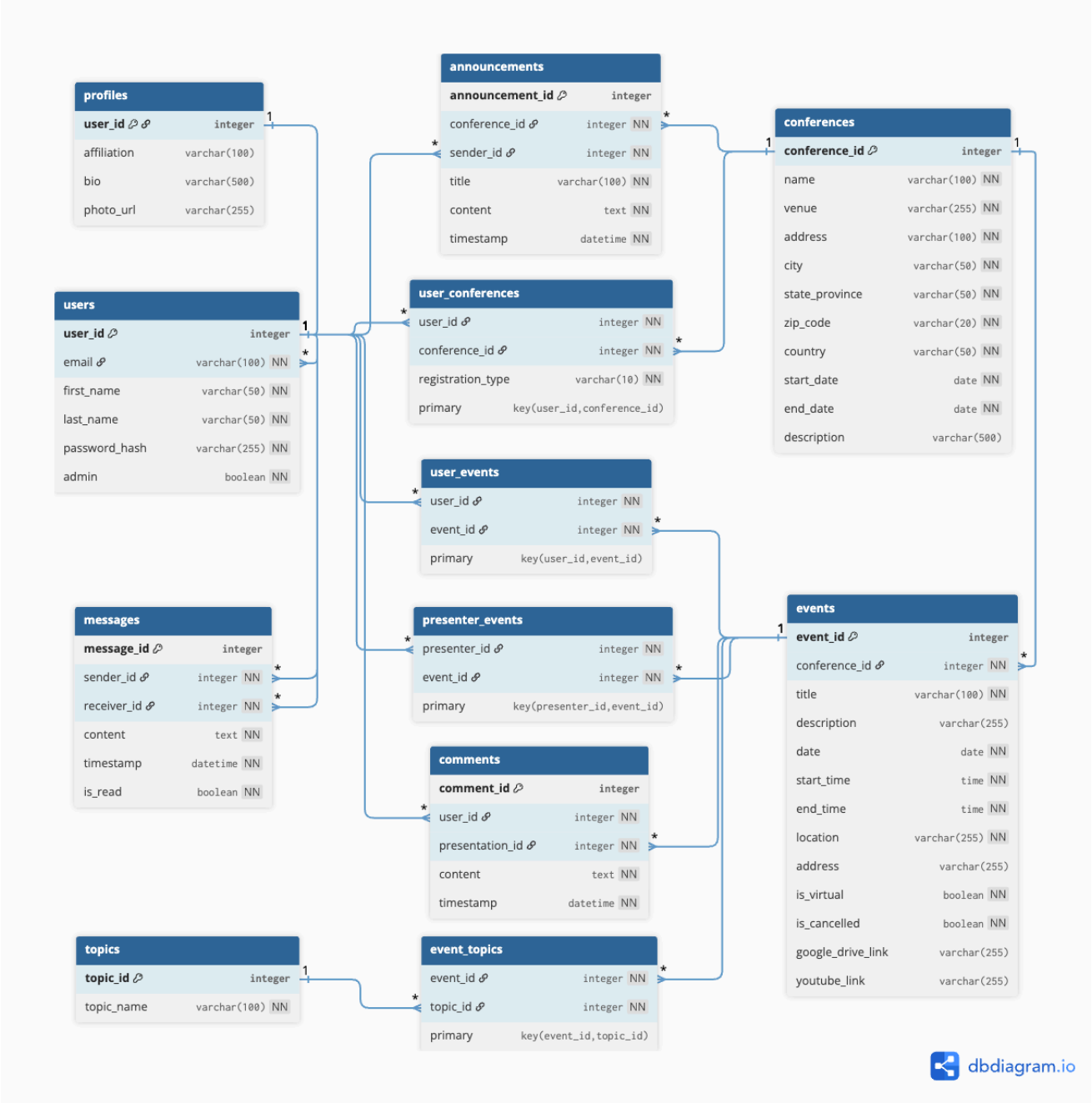


Figure 3.1 Entity-Relationship Diagram for the IWAC Database

The following tables (Table 3.1.1 through Table 3.1.12) 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
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	Indicates if the user has registered for this conference in person or online.	Yes

Table 3.1.5 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 the event.	Yes

start_time	TIME	-	Start time of the event.	Yes
end_time	TIME	-	End time of the event.	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 3.1.6 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 3.1.7 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 3.1.8 Comments Table Definition

Field	Type	Size	Description	Required
comment_id	INT	-	Auto-incremented unique identifier for each comment.	Yes
user_id	INT	-	References users(user_id). The user who sent the comment.	Yes
event_id	INT	-	References events(event_id). The event being commented on.	Yes
content	VARCHAR	1000	Text content of the comment.	Yes
timestamp	DATETIME	-	Date and time when the comment was posted.	Yes

Table 3.1.9 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 3.1.10 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

Table 3.1.11 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 3.1.12 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

Table 3.2.1 and Table 3.2.2 describe the files used in the IWAC system, namely the banner images that are used for each conference and the profile pictures that users may optionally upload. Each file attribute is explained with its name, data type, size, description, and optionality.

Table 3.2.1 Banner Image File Description

Field	Type	Size	Description	Required
file_name	VARCHAR	255	Actual file name stored in the server directory.	Yes
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	-	Size of the file in bytes.	Yes
width	INT	-	Image width in pixels.	Yes
height	INT	-	Image height in pixels.	Yes

Table 3.2.2 Profile Picture File Description

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
width	INT	-	Image width in pixels.	Yes
height	INT	-	Image height in pixels.	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 fulfilled by at least one method ensures that we satisfy all Functional Requirements in the development of the IWAC application. These methods, and the objects that implement them are diagramed in Figure 2.2.

Table 4 Requirements Matrix Table

Functional Requirement 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: _____ Date: 11/17/2025

Signature: _____ Date: 11/17/2025

Signature: _____ Date: 11/17/2025

Signature: _____ Date: 11/17/2025

Signature: _____ Date: 11/17/2025

Signature: _____ Date: 11/20/2025

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:  Date: 11/17/2025

Comments: _____

_____

Brett Palmer

Signature:  Date: 11/17/2025

Comments: _____

_____

Monica Agneta

Signature:  Date: 11/17/2025

Comments: _____

_____

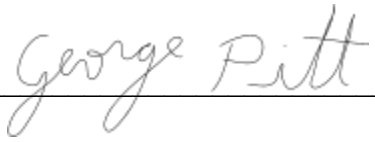
Ben Caras

Signature:  Date: 11/17/2025

Comments: _____

_____

George Pitt

Signature:  Date: 11/17/2025

Comments: _____

_____

Appendix C – Document Contributions

Brett

Brett designed the database structure and represented it in diagram 3.1, created tables 3.1.1 through 3.1.8, and wrote the text within section 3.1. He did general editing for consistent formatting and grammar within the document.

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 3.1.9 through 3.1.12, 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 diagrams associated with this section. Ben also wrote Appendix A.

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