CASA Donation Management Web Application

Design Paper

Presented by: Ethan Bostick, Joshua Byrne, James Heathcock, Garrett Mckenzie, Maximilian Redman, and Carter Walker

Rappahannock Area Court Appointed Special Advocates
509 C Lafayette Blvd, Fredericksburg, VA 22401
(540) 710-6199

Department of Computer Science, University of Mary Washington

CPSC 430: Software Engineering

Dr. Jennifer A. Polack

November 12, 2025

This page is left intentionally blank.

Table of Contents

1 Introduction

1.1 Statement of Purpose

This software design document describes the overall architecture and system design of the CASA Donation Management Web Application. The purpose of this document is to outline the system's key structural and functional components, showing how the front-end, backend, and database integrate to support CASA's donation and event management operations.

This document details the core subsystems, including donation and fundraising tracking, event management, data analysis and reporting, file-based data import/export, and automated email generation. It also defines how these components interact to create a cohesive and user-friendly system for Rappahannock CASA staff. For readers without a technical background, Section 6 provides definitions of technical terms used throughout this paper.

1.2 Intended Audience

This document is intended for the developers of the CASA Donation Management Web Application, as well as Edie Evans, the Executive Director of Rappahannock CASA. Additional readers include CASA staff who will use or maintain the system, and University of Mary Washington faculty overseeing the project. The content assumes some familiarity with system development concepts but is written to be accessible to all key stakeholders.

1.3 Scope

The CASA Donation Management Web Application is designed for internal use by Rappahannock CASA staff. It streamlines the process of managing donations, organizing fundraising events, and communicating with donors.

Key system functions include:

- A donation and event tracking system, enabling staff to manage donor activity and fundraising progress.
- A data storage subsystem with file-based import/export functionality for quick updates and report generation.
- A calendar-based event manager to create, edit, and track upcoming and past events.
- A dynamic reporting suite that generates data visualizations and summaries based on stored information.
- An automated email subsystem that can send targeted messages—such as alerts or thank-you notes—to selected donors or event participants.

By integrating these features, the application significantly enhances CASA's operational efficiency, allowing staff to seamlessly generate reports, maintain donor records, and manage events from one centralized platform.

1.4 References

- Rappahannock CASA Homepage: https://rappahannockcasa.org/

1.5 Overview

The remainder of this document outlines the system's detailed design and data structures. It includes architectural overviews, entity-relationship diagrams, and component descriptions explaining how each subsystem functions and interacts with others. The paper is organized into five main sections:

1. Introduction
2. System Overview
3. Data Design
4. Component Design
5. Requirements Matrix

2 System Overview

2.1 High-Level Design

This Level 0 Data Flow Diagram shows how the CASA System works and how data moves between the CASA staff, the system processes, and the main database (CASADB). The diagram helps explain how the system handles logins, event management, reporting, and email communication in an organized way.

The system includes several main processes that connect the CASA staff to the database:

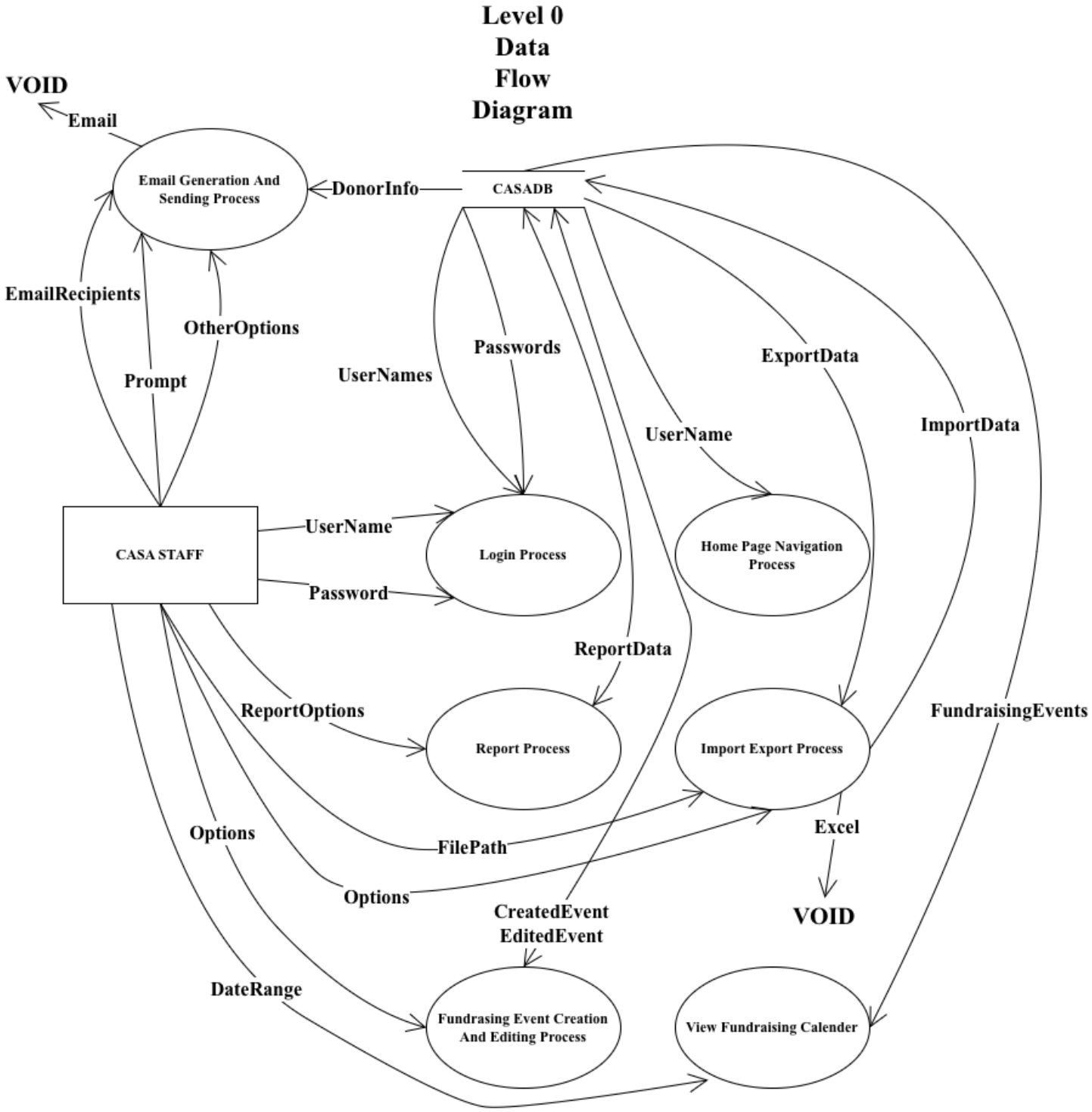
- Login Process – Verifies the staff's username and password with the CASADB to make sure only authorized users can access the system.
- Home Page Navigation Process – Lets users move between different parts of the system after logging in, helping them find and access the information or tools they need.
- Import/Export Process – Allows staff to bring in data from external files or export data (like reports or event information) to Excel for easy sharing and record keeping.
- Report Process – Generates reports based on selected options. These reports give staff useful insights into donor details, events, and other important information stored in the CASADB.
- Fundraising Event Creation and Editing Process – Lets staff create new fundraising events or update existing ones. The system saves these events in the database so they can be viewed later in the calendar.

- Email Generation and Sending Process – Helps staff send out emails to donors or other recipients. Staff can choose recipients and message options, and the system sends the emails automatically.

At the center of everything is CASADB, which stores all important information, such as donor data, staff login details, event records, and report data. CASA staff are the main users who provide input (like login credentials, report options, and event details) and receive outputs (such as reports, emails, and calendar updates).

Overall, this diagram shows how the CASA System brings together staff interactions, data management, and communication processes to keep operations smooth, organized, and efficient.

Note: Figure is displayed on the following page.


Level 0
Data
Flow
Diagram
VOID
Email
Email Generation And Sending Process
DonorInfo
CASADB
EmailRecipients
OtherOptions
Prompt
UserNames
Passwords
ExportData
ImportData
UserName
CASA STAFF
UserName
Password
Login Process
Home Page Navigation Process
ReportData
ReportOptions
Report Process
Import Export Process
FundraisingEvents
Options
FilePath
Excel
Options
CreatedEvent
EditedEvent
VOID
DateRange
Fundrasing Event Creation And Editing Process
View Fundraising Calender

2.2 Low-Level Design

The Low-Level Design section provides a detailed representation of how individual components of the CASA system operate and interact through Data Flow Diagrams (DFDs). These diagrams break down the system's internal processes, data stores, and external entities, showing how data moves and is transformed within the system. Each DFD corresponds to a specific functional area, such as user login, report generation, fundraising event management, or automated email communication, and builds upon the high-level design to illustrate the underlying data exchanges and logic.

By examining these diagrams, we gain a clear understanding of how data is collected, processed, stored, and output across different parts of the system. This helps verify that each process performs its intended role, interfaces correctly with related components, and maintains consistency with the overall system requirements. The Low-Level Design ensures the system's workflow is logically structured, supports scalability, and aligns with CASA's operational goals of efficiency, accuracy, and automation.

2.2.1 Report Process

This Level 1 Data Flow Diagram illustrates the report generation and analysis subsystem within the CASA platform. It focuses on how the Analyst Staff interacts with the system to generate, view, and store analytical reports derived from organizational data.

The subsystem is composed of two main processes:

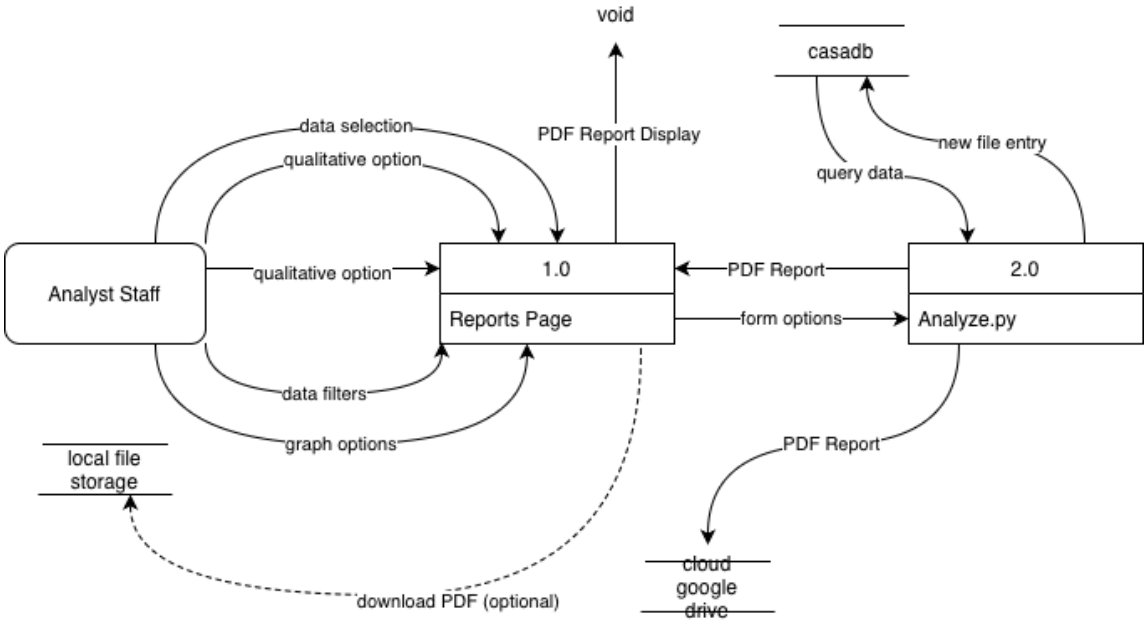
- Reports Page (1.0) – Serves as the user interface for report creation. Analyst staff members fill out the form on this page to specify data filters, qualitative or graphing options, and report parameters. The page sends these options to the backend analysis script and, upon receiving the processed data, displays the resulting PDF report. Users also have the option to download the report locally for recordkeeping or further analysis. This centralized form design was chosen for flexibility, allowing report criteria to evolve without altering system architecture.

- Analyze.py (2.0) – Handles the back-end data processing and report generation. It receives form options from the Reports Page and queries the CASADB for relevant data from tables such as dbEvents, donations, donor, and associations. Using Python libraries like pandas, seaborn, and matplotlib, the script processes and visualizes data, generates a PDF report, and updates the files and reports tables in CASADB to store metadata and access paths. The system also uploads the finished report to a connected cloud drive (e.g., Google Drive) for external access and redundancy.

Design decisions and tradeoffs: The integration of Python for analysis was chosen for its flexibility, mature data-processing ecosystem, and ease of generating high-quality visualizations. Alternatives such as built-in SQL reporting or web-based analytics frameworks were considered but rejected to maintain a lightweight and customizable architecture. Using a centralized CASADB ensures data consistency and supports future scalability for additional report types or visualization styles.

Overall, this design emphasizes modularity, reproducibility, and accessibility, ensuring analysts can efficiently generate and distribute data-driven reports while maintaining synchronized records within the CASA database.

Note: Figure is displayed on the following page



2.2.2 Email Generation and Sending Process

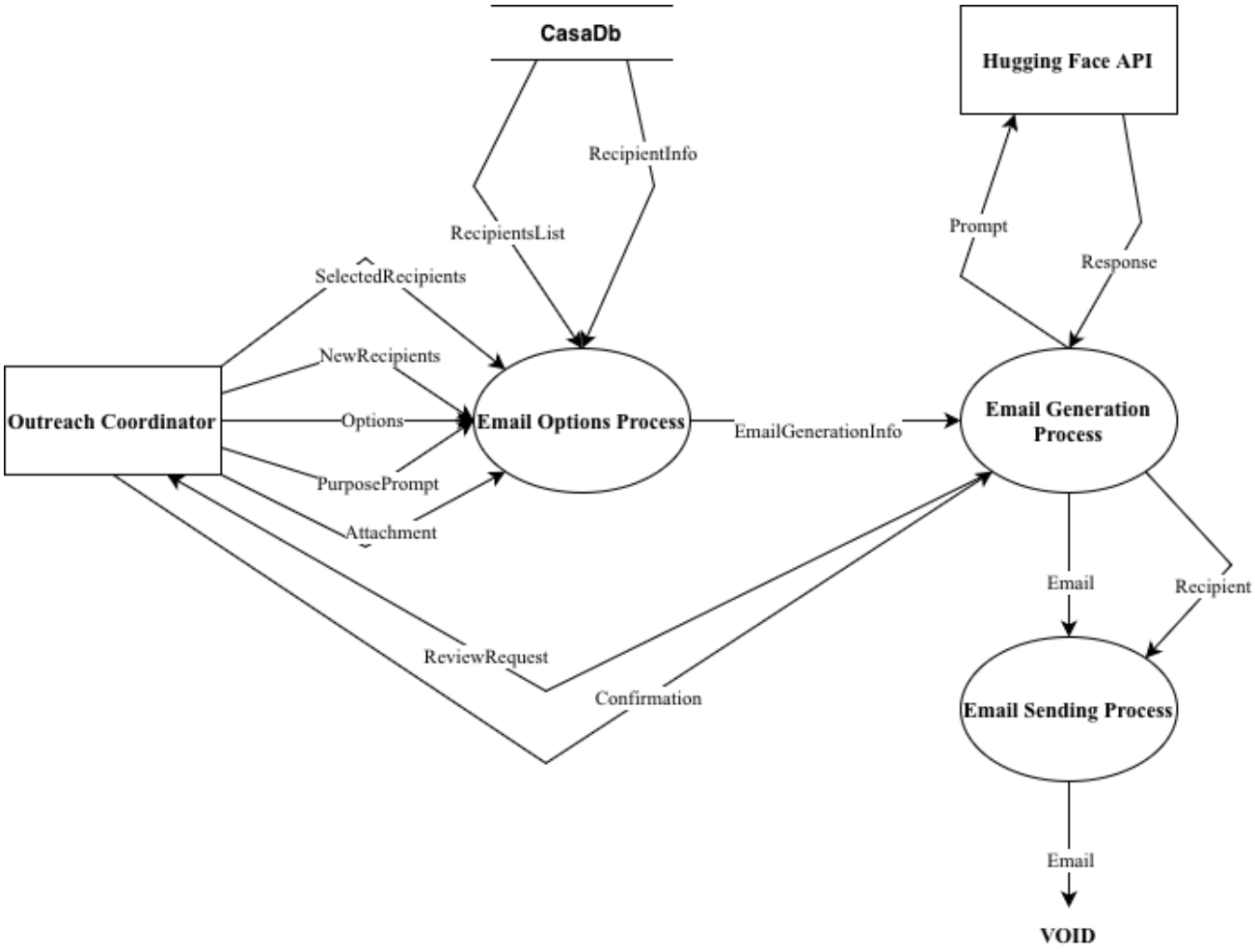
This Level 1 Data Flow Diagram illustrates the CASA system's automated email generation and sending process. It shows how the Outreach Coordinator interacts with the CASADB and the Hugging Face API to configure, generate, and deliver personalized emails to donors and community members.

The system is built around three key processes:

- Email Options Process – Allows the user to select recipients, attach files, and define the purpose of the email. This design consolidates configuration into one process for efficiency, avoiding the added complexity of multiple setup modules.
- Email Generation Process – Uses the Hugging Face API to create tailored email content based on prompts and recipient data. The API was chosen over a self-hosted model to reduce infrastructure costs and simplify maintenance, trading off some control for scalability and ease of integration.
- Email Sending Process – Sends the generated emails to selected recipients. The system supports both individualized and bulk sending, offering flexibility while maintaining personalization when needed.

These design choices prioritize simplicity, automation, and adaptability. Alternatives such as internal model hosting or segmented configuration workflows were considered but dismissed due to increased complexity and maintenance overhead. The resulting design achieves a balanced tradeoff between personalization, system efficiency, and long-term scalability.

Note: Figure is displayed on the following page



2.2.3 Home Page Navigation Process

This Level 1 Data Flow Diagram illustrates how the system's homepage manages user navigation, session handling, and page routing within the donor management application. It highlights how the User, Homepage, and index.php process

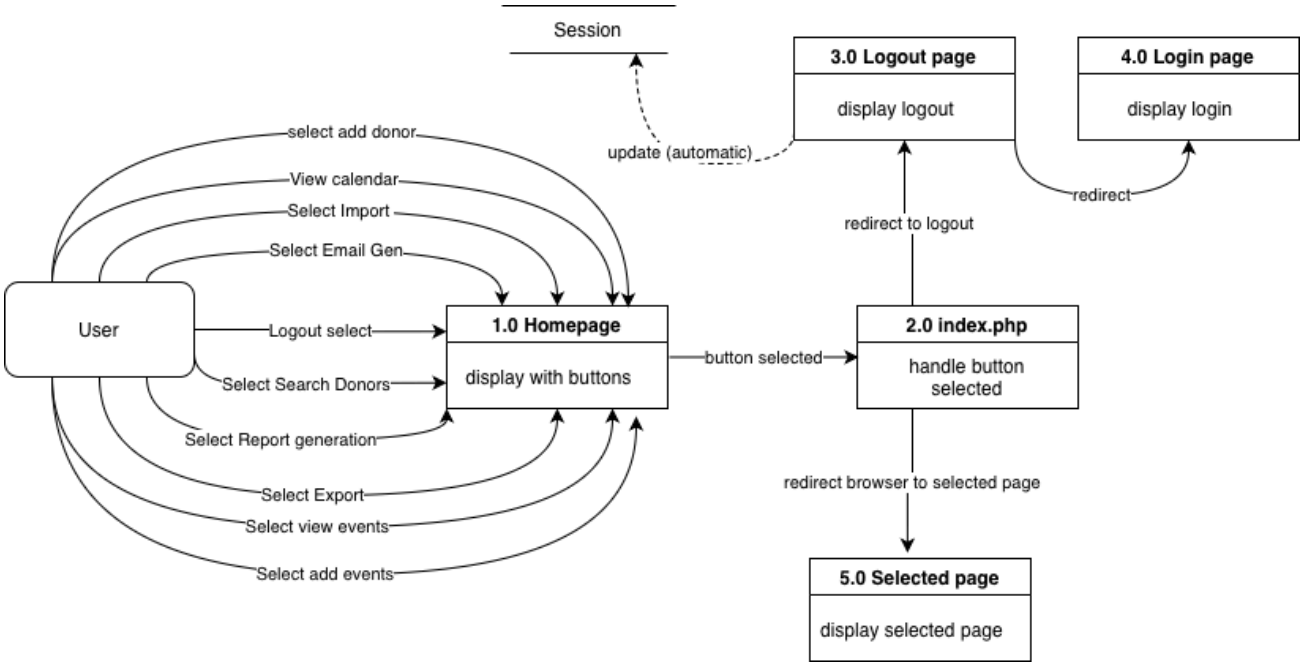
interact to control transitions between modules such as donor management, report generation, event scheduling, and system login/logout.

The design centers on three main processes:

- Homepage Display Process – Presents the user with a central interface of interactive buttons to access all major system functions. This design consolidates user interaction into a single entry point, improving accessibility and reducing interface complexity.
- Routing Process (index.php) – Acts as the system's control hub, interpreting which button the user selects and redirecting them to the appropriate module or logout page. This process maintains logical separation between navigation control and page content, supporting modular updates and scalability.
- Session and Authentication Process – Manages session data to ensure users remain authenticated as they navigate through pages. When logout is triggered, the session automatically terminates, redirecting the user to the login page for re-entry.

These design decisions streamline navigation and maintain secure session handling while minimizing redundant logic across modules. The result is a lightweight, user-friendly homepage structure that centralizes interaction, promotes modular maintainability, and supports consistent system flow.

Note: Figure is displayed on the following page

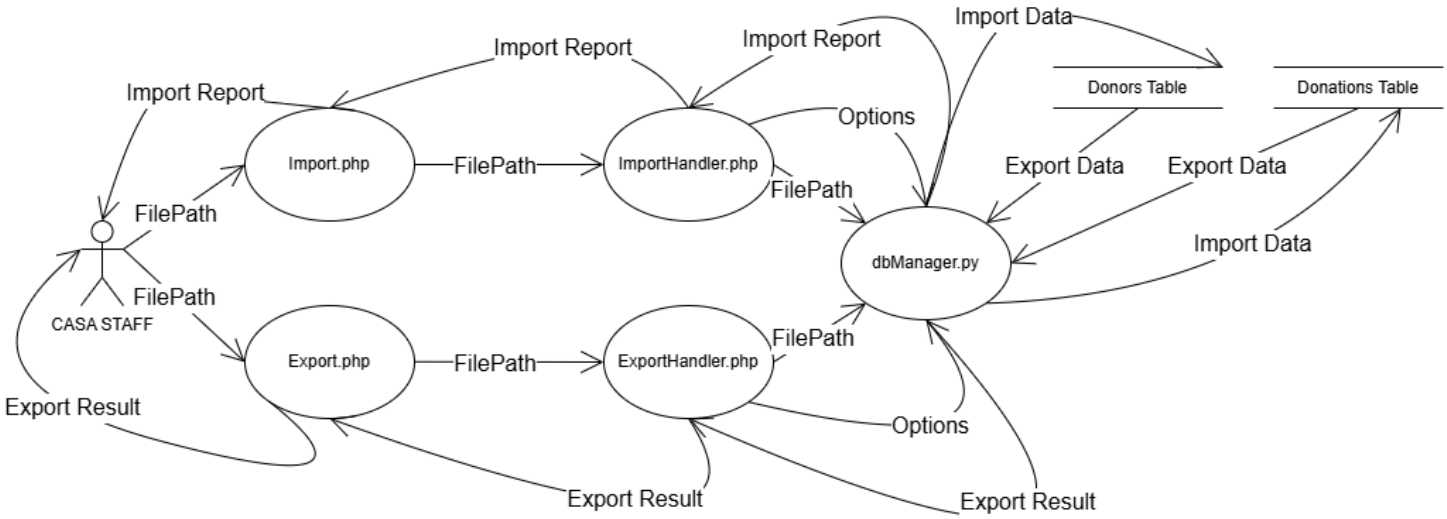


2.2.4 Import and Export Process

This Level 1 Data Flow Diagram illustrates the workflow of a data import/export system for CASA staff. Users provide a file path to initiate either an import or export process. For imports, import.php passes the file path to importHandler.php, which interacts with dbManager.py to insert data into the dbDonors and dbDonations tables. For exports, export.php sends the file path to exportHandler.php, which retrieves data from the same tables via dbManager.py. The system then generates reports or results that are sent back to the CASA staff user.

This design enables reliable two-way data flow, simplifying recordkeeping and backups.

Note: Figure is displayed on the following page.



2.2.5 Fundraising Event Creation and Editing Process

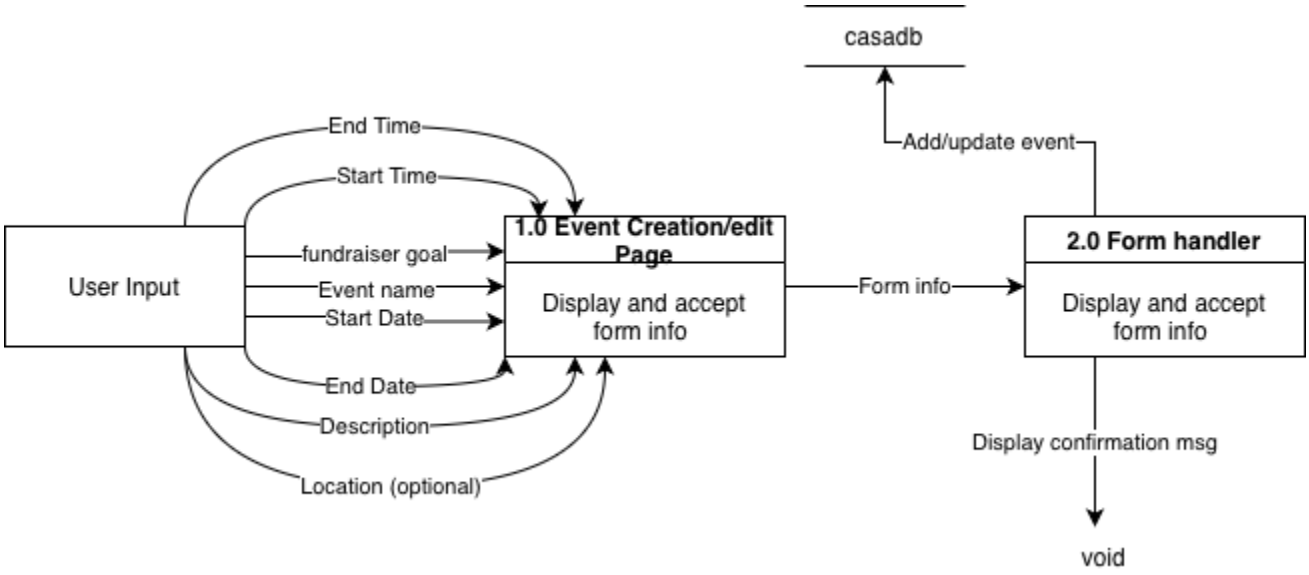
This Level 1 Data Flow Diagram illustrates the process by which users create or edit events within the CASA system. It details how user inputs flow through the event creation interface and are processed by the form handler before being stored in the database.

The design centers on two key processes:

- Event Creation/Edit Process – Allows users to enter or modify event details such as name, dates, times, fundraiser goal, description, and optional location. This page consolidates all relevant fields into a single, guided form interface, minimizing user confusion and promoting data completeness.
- Form Handling Process – Validates and processes the submitted event information, updating CASADB with new or modified event records. Once the operation is complete, the system provides a confirmation message to the user, ensuring transparency and feedback for each submission.

This design prioritizes clarity, simplicity, and reliability. By separating user-facing input from backend data handling, the system maintains strong modularity and reduces the potential for data entry errors, while ensuring that updates to event data remain consistent and secure within the database.

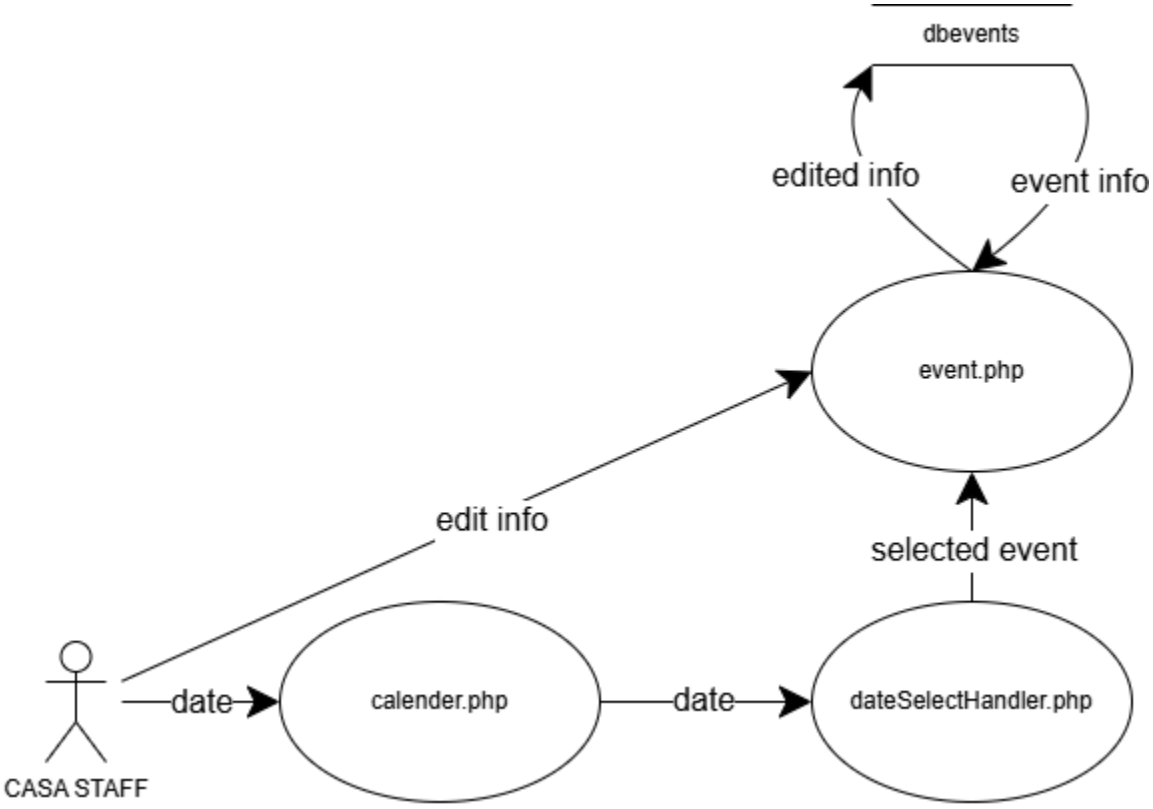
Note: Figure is displayed on the following page.



2.2.6 View Fundraising Calendar Process

This diagram shows the data flow for viewing and editing events in the CASA staff system. The process begins when CASA staff select a date in calendar.php, which passes the selected date to dateSelectHandler.php. The handler retrieves the corresponding event data and sends it to event.php, where event details are displayed. CASA staff can then edit the event information, which is updated in the database dbevents and reflected back in event.php. The system supports both viewing and modifying event details for selected dates.

Note: Figure is displayed on the following page



2.2.7 Login Process

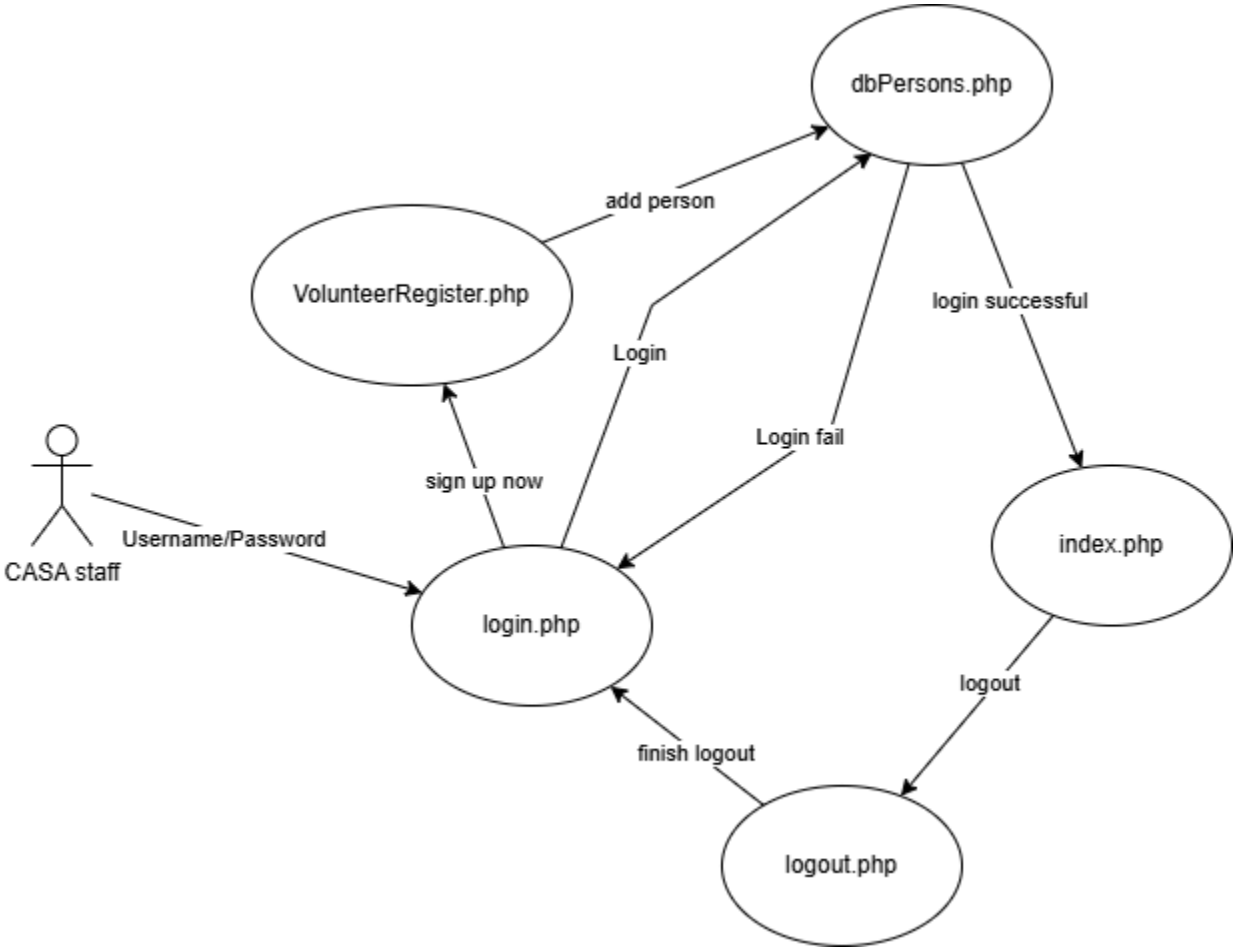
This Level 1 Data Flow Diagram illustrates the login and authentication process for CASA staff. It shows how users interact with the login interface, backend handler, and CASADB to verify credentials and manage sessions.

The design centers on two key processes:

- Login Page Process – Allows users to enter credentials (username and password). The form submits data to the backend for validation.
- Authentication Handler Process – Hashes the password, queries CASADB for a match in dbPersons, and creates a session if valid. It handles redirects to the homepage on success or error messages on failure.

This design ensures secure access control and smooth transitions, with alternatives for invalid inputs or database errors.

Note: Figure is displayed on the following page.



3 Data Design

3.1 Data Description

3.1.1 Database Description

The database used for this system is SQL-based, designed to efficiently store, manage, and retrieve data related to donors, events, reports, and communications. Its structure supports the coordination of fundraising activities, donor engagement, and administrative oversight. The system enables administrators to manage events, track donations, generate reports, and ensure all necessary communications (such as thank-you or alert emails) are properly sent to donors.

The database is tightly integrated with the front end, allowing users to access and manipulate relevant information with minimal steps. Each table within the schema plays a key role in ensuring data consistency and accessibility across the system. The following section describes the primary tables and their purposes.

Table: dbPersons

- **Purpose:** Stores login credentials and access-level information for individuals interacting with the system.
- **Fields:**
    - id (text, PK): Unique identifier for each user.
    - password (text): Encrypted user password.
    - name (text): User's name.
    - accessLevel (int): Determines the user's permissions within the system.
- **Relationships:** Acts independently but provides administrative and authentication context for other operations.

Table: dbEvents

- **Purpose:** Tracks fundraising events, including their goals, timeframes, and progress.
- **Fields:**
    - id (int, PK): Unique event ID.
    - name (text): Event name.
    - goalAmount (float): Target fundraising goal.
    - startDate / endDate (DateTime): Defines the event duration.
    - startTime / endTime (char[5]): Time fields for event scheduling.
    - description (text): Summary of the event.
    - completed (int): Indicator for event completion status.
    - location (text): Event location.
- **Relationships**: Connected to donations and associations, linking events to donors, donations, and emails.

Table: donations

- **Purpose:** Records all individual donations made by donors.
- **Fields:**
    - id (int, PK): Unique donation ID.
    - amount (float): Amount donated.
    - reason (text): Reason or campaign purpose.
    - date (DateTime): Date of donation.
    - fee (float): Processing fee.
    - thanked (int): Indicator if a thank-you email has been sent.
- **Relationships**: Linked to dbEvents and donors through associations.

Table: Donors

- **Purpose:** Stores donor personal and contact information for communication and recordkeeping.
- **Fields:**
    - id (int, PK): Unique donor ID.
    - first, last (text): Donor name.
    - email (text): Donor contact email.

  - address fields: street, city, state, zip.
  - phone, gender, notes.
- **Relationships**: Connects to donations, emails, and events via associations.

Table: Emails

- **Purpose:** Contains email message templates and their associated purposes.
- **Fields:**
  - id (int, PK): Unique email ID.
  - message (text): Body of the email.
  - purpose (text): Purpose (e.g., thank-you, alert).
- **Relationships**: Linked through the associations table to track which emails have been sent to which donors for specific events.

Table: Reports

- **Purpose:** Stores generated report metadata for performance tracking and analytics.
- **Fields:**
  - id (int, PK): Unique report ID.
  - type (text): Type of report (e.g., summary, donation trends).
  - dateRangeStart / dateRangeEnd (DateTime): Range of data covered.
  - name (text): Report name.
- **Relationships**: Connected to files via associations for external data storage and export references.

Table: Files

- **Purpose:** Manages links to files stored locally or in external drives.
- **Fields:**
  - filePath (text, PK): Local file path.
  - linkToDrive (text): Cloud storage link (e.g., Google Drive).
- **Relationships**: Connected to reports and other entities through associations.

Table: Associations

- **Purpose:** A 4-nary relationship table that connects dbEvents, donations, donors, emails, and files. It allows the system to determine complex relationships—such as whether all donors to a specific event have received a related thank-you email and if reports or files are linked to that communication.
- **Fields:**
  - eventID, donationID, donorID, emailID, filePath.
- **Relationships:** Serves as a central link between all major entities, enabling robust data queries and traceability across system operations.

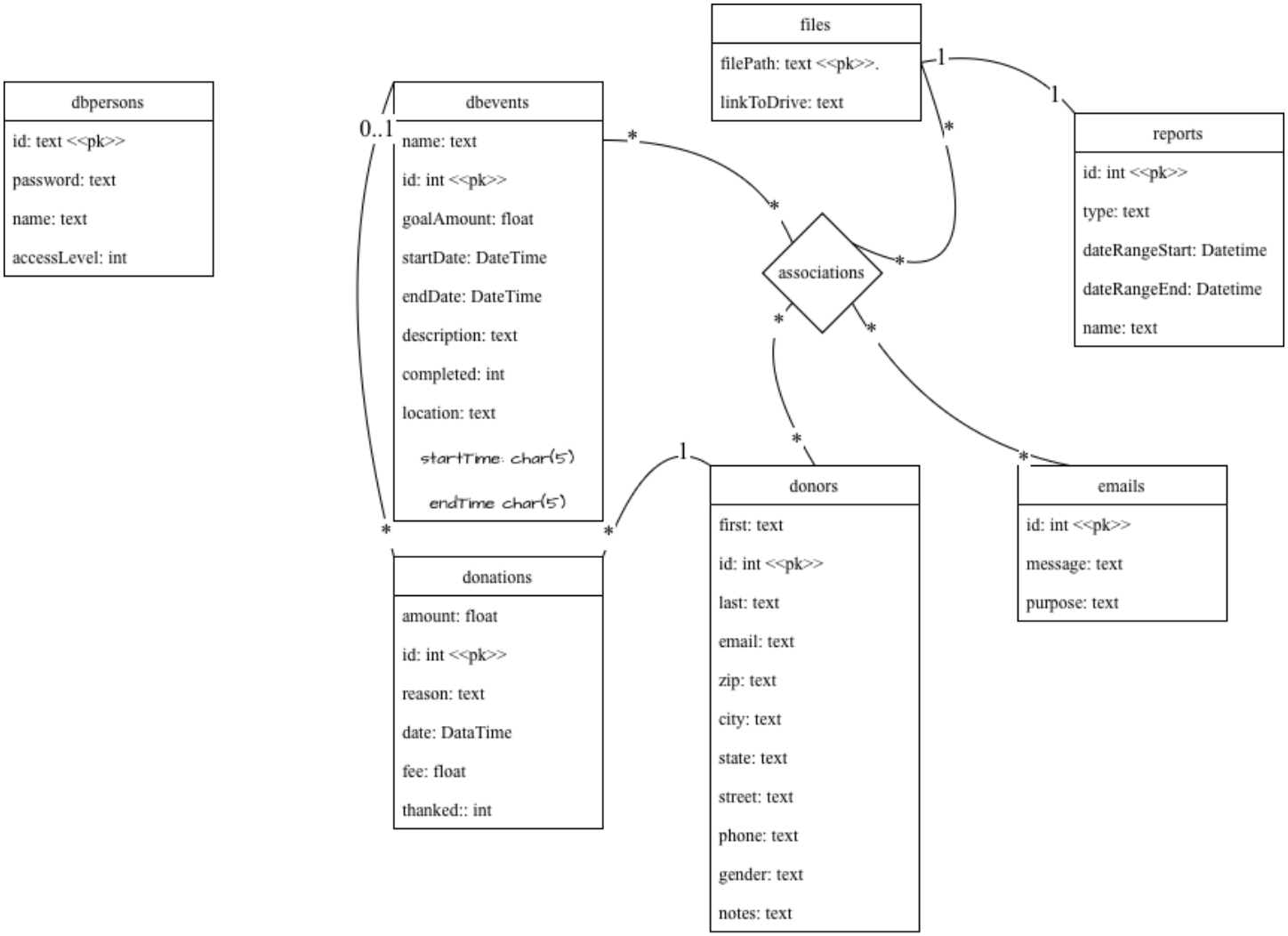
3.1.2 Entity Relationship Diagram

This Entity–Relationship Diagram defines the core data model for the system, illustrating how donors, events, donations, emails, and reports are interconnected. The schema is designed to support efficient tracking of fundraising activities and communication history. A key component is the associations entity, which functions as a 4-nary relationship connecting events, donors, donations, emails, and files. This enables complex queries such as determining whether a thank-you or alert email has been sent to every donor who contributed to a specific event.

Each table is purposefully normalized:

- dbpersons manages user authentication and access control.
- dbevents tracks fundraising events and their details.
- donations records individual contributions linked to events and donors.
- emails stores message content and purpose for communication tracking.
- files and reports support documentation, analytics, and external file linkage.

The design prioritizes clarity and traceability, balancing normalization with query efficiency to support reporting and data integrity throughout the system.



3.2 Data Dictionary

A data dictionary defines the structure and relationships of all entities within the system, outlining how data objects interact and support key functionality. This database manages events, donations, donors, and communication records, ensuring that every email or alert sent can be traced to a specific donor, donation, and event.

Classes/Entities: dbEvents

Attributes

- eventID (PK) – INT
- name – TEXT
- goalAmount – FLOAT
- startDate / endDate – DATETIME

- startTime / endTime – CHAR(5)
- description – TEXT
- completed – INT
- location – TEXT

Functions

- add_event(event)
- update_event(eventID, details)
- delete_event(eventID)
- get_event(eventID)
- get_all_events()

GUI Interactions

- Create and manage events
- View event details and participants

Classes/Entities: dbDonors

Attributes

- donorID (PK) – INT
- first, last – TEXT
- email – TEXT
- street, city, state, zip – TEXT
- phone – TEXT
- gender – TEXT
- notes – TEXT

Functions

- add_donor(donor)
- update_donor(donorID, details)
- get_donor(donorID)
- get_all_donors()

GUI Interactions

- Manage donor records
- View donation and email history

Classes/Entities: donations

Attributes

- donationID (PK) – INT
- donorID (FK) – INT
- eventID (FK) – INT

- amount – DECIMAL(10,2)
- reason – TEXT
- date – DATETIME
- fee – FLOAT
- thanked – INT

Functions

- add_donation(donation)
- update_donation(donationID, details)
- get_donation(donationID)
- get_donations_by_event(eventID)

GUI Interactions

- Record new donations
- Generate donation summaries

Classes/Entities: dbEmails

Attributes

- emailID (PK) – INT
- message – TEXT
- purpose – TEXT

Functions

- create_email(email)
- send_email(emailID, donorID)
- get_email(emailID)
- get_emails_by_type(type)

GUI Interactions

- Compose and send emails
- Track communication history

Classes/Entities: associations

Attributes

- eventID (FK) – INT
- donationID (FK) – INT
- donorID (FK) – INT
- emailID (FK) – INT
- filePath – TEXT

Description
This 4-nary association links events, donations, donors, and emails. It enables queries such as:

> "Has a thank-you or alert email associated with a specific event been sent to every donor who donated to that event?"

Functions

- link_association(eventID, donationID, donorID, emailID, filePath)
- get_association(eventID, donorID)
- check_email_sent(eventID, donorID)

GUI Interactions

- Generate event-specific donor communication reports
- Verify completion of thank-you emails

4 Component Design

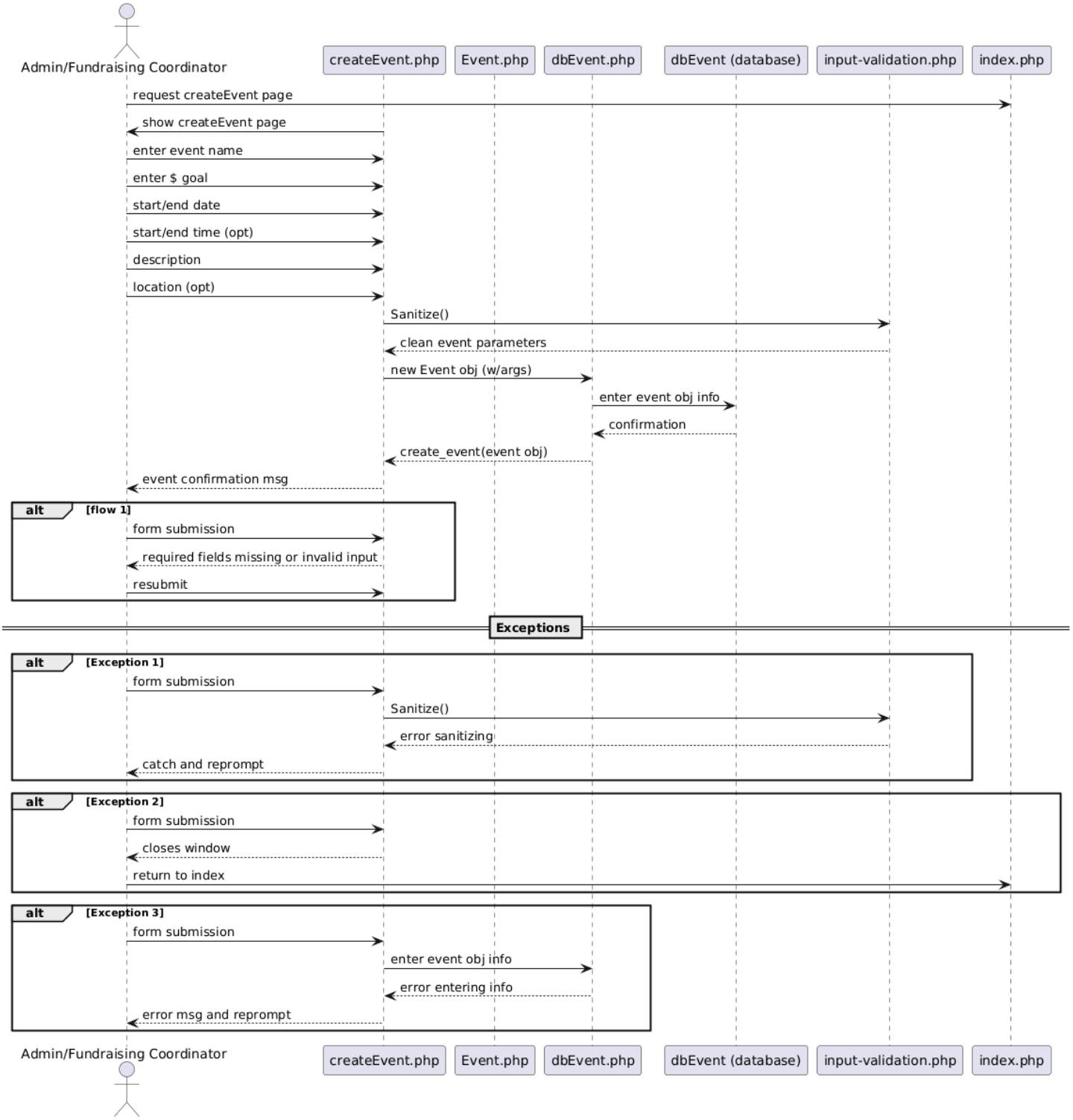
4.1 Sequence Diagrams

Sequence diagrams illustrate how different parts of the system communicate over time to complete a process. They capture the order of interactions between users, system components, and the database, providing a clear picture of how data and control flow through the application. These diagrams help developers understand the system's dynamic behavior and ensure that component interactions align with design expectations. The following sequence diagrams represent key workflows implemented during the initial development sprints.

4.1.1 Create Fundraising Event

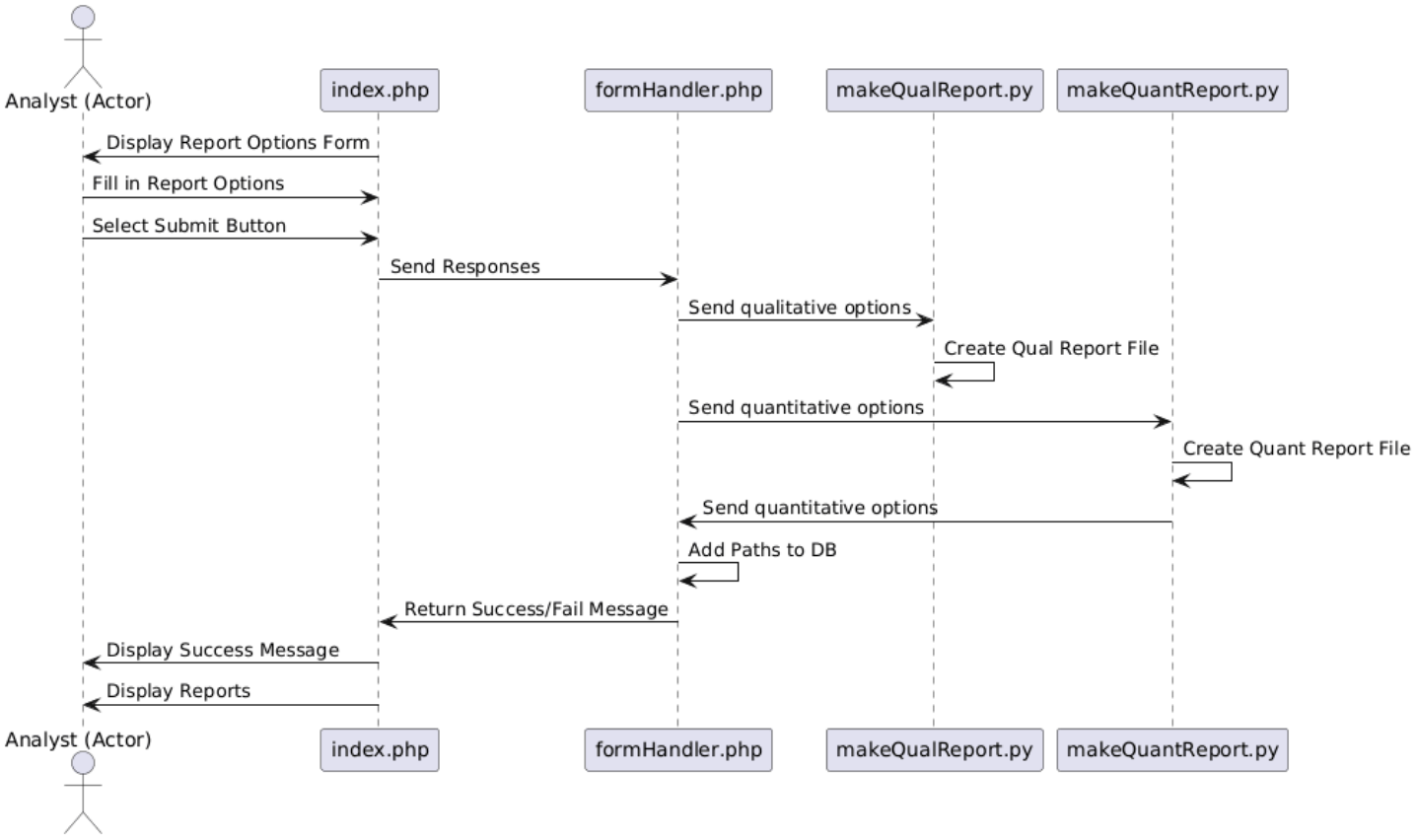
This sequence diagram shows the process for an admin or fundraising coordinator creating a new event within the system. The user accesses the createEvent.php page and inputs all required event details such as the name, goal amount, start and end dates, times, description, and optional location. The system sanitizes the input data through input-validation.php before creating a new Event object. The validated event information is then sent to dbEvent.php, where it is stored in the database. Upon successful creation, a confirmation message is displayed. The diagram also includes exception and alternate flows to handle invalid inputs, form resubmissions, or database entry errors.

Note: Figure is displayed on the following page


Admin/Fundraising Coordinator
createEvent.php
Event.php
dbEvent.php
dbEvent (database)
input-validation.php
index.php
request createEvent page
show createEvent page
enter event name
enter $ goal
start/end date
start/end time (opt)
description
location (opt)
Sanitize()
clean event parameters
new Event obj (w/args)
enter event obj info
confirmation
create_event(event obj)
event confirmation msg
alt [flow 1]
form submission
required fields missing or invalid input
resubmit
Exceptions
alt [Exception 1]
form submission
Sanitize()
error sanitizing
catch and reprompt
alt [Exception 2]
form submission
closes window
return to index
alt [Exception 3]
form submission
enter event obj info
error entering info
error msg and reprompt
Admin/Fundraising Coordinator
createEvent.php
Event.php
dbEvent.php
dbEvent (database)
input-validation.php
index.php

4.1.2 Report Generation

This sequence diagram illustrates the process of report generation within the system. The analyst begins by selecting report parameters on the index.php page and submitting the form. The formHandler.php script processes the request and sends the selected qualitative and quantitative options to their respective Python modules. The makeQualReport.py module generates the qualitative report, while the makeQuantReport.py module creates the quantitative report. Once both reports are created, their file paths are added to the database. The system then returns a success or failure message to the user, displaying the resulting reports upon successful completion.

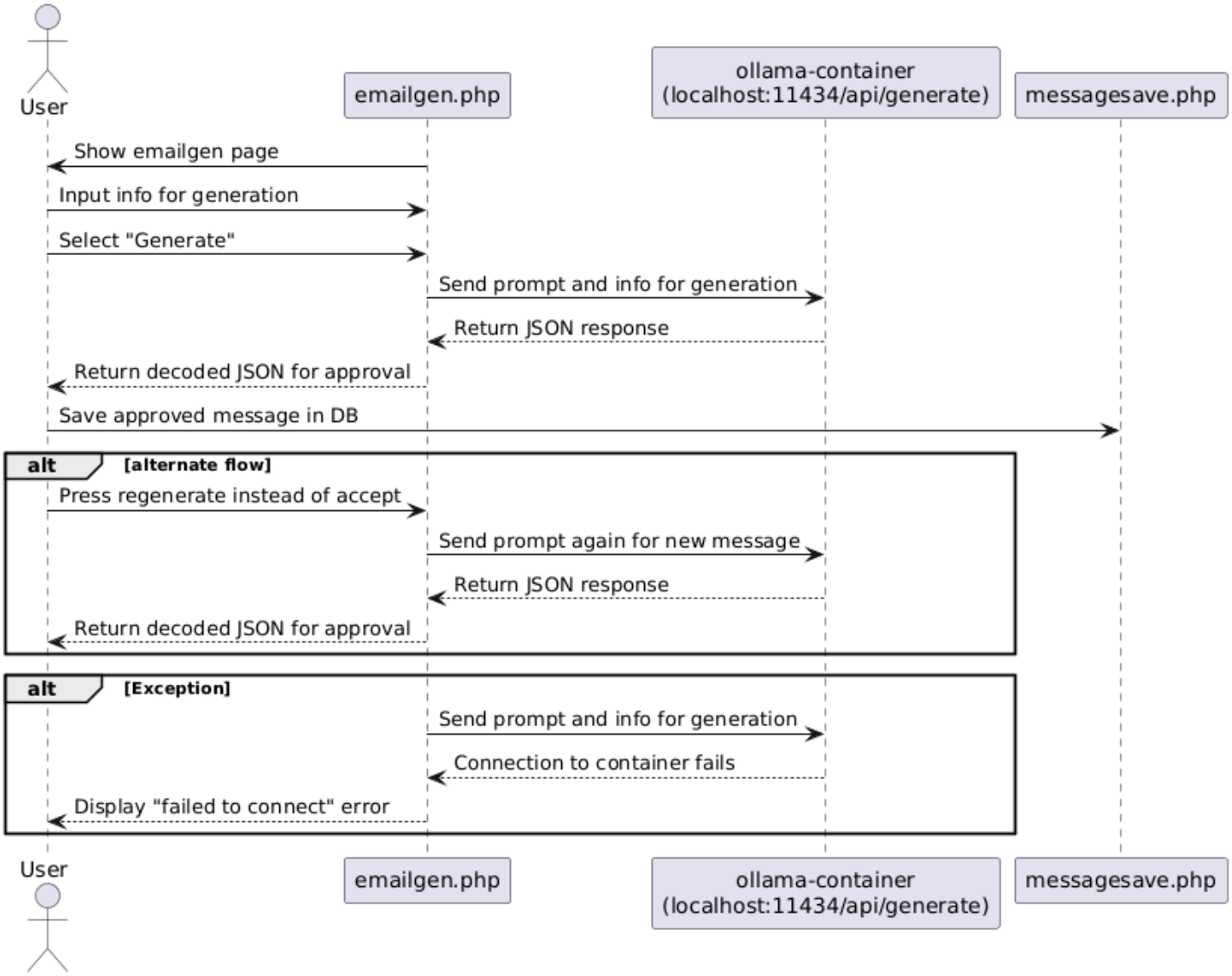


4.1.3 Email Generation

This sequence diagram illustrates the process of AI-generated email creation within the system. The user accesses emailgen.php, enters the necessary input for the email, and selects “Generate.” The system sends the provided prompt and parameters to the Hugging Face API for generation. Once the response is returned in JSON format, it is decoded and displayed to the user for review and approval. Upon acceptance, the approved message is saved to the database through messagesave.php.

The alternate flow occurs when the user chooses to regenerate the message instead of approving it. In this case, the same prompt is sent again to the API, which returns a new generated response for user approval.

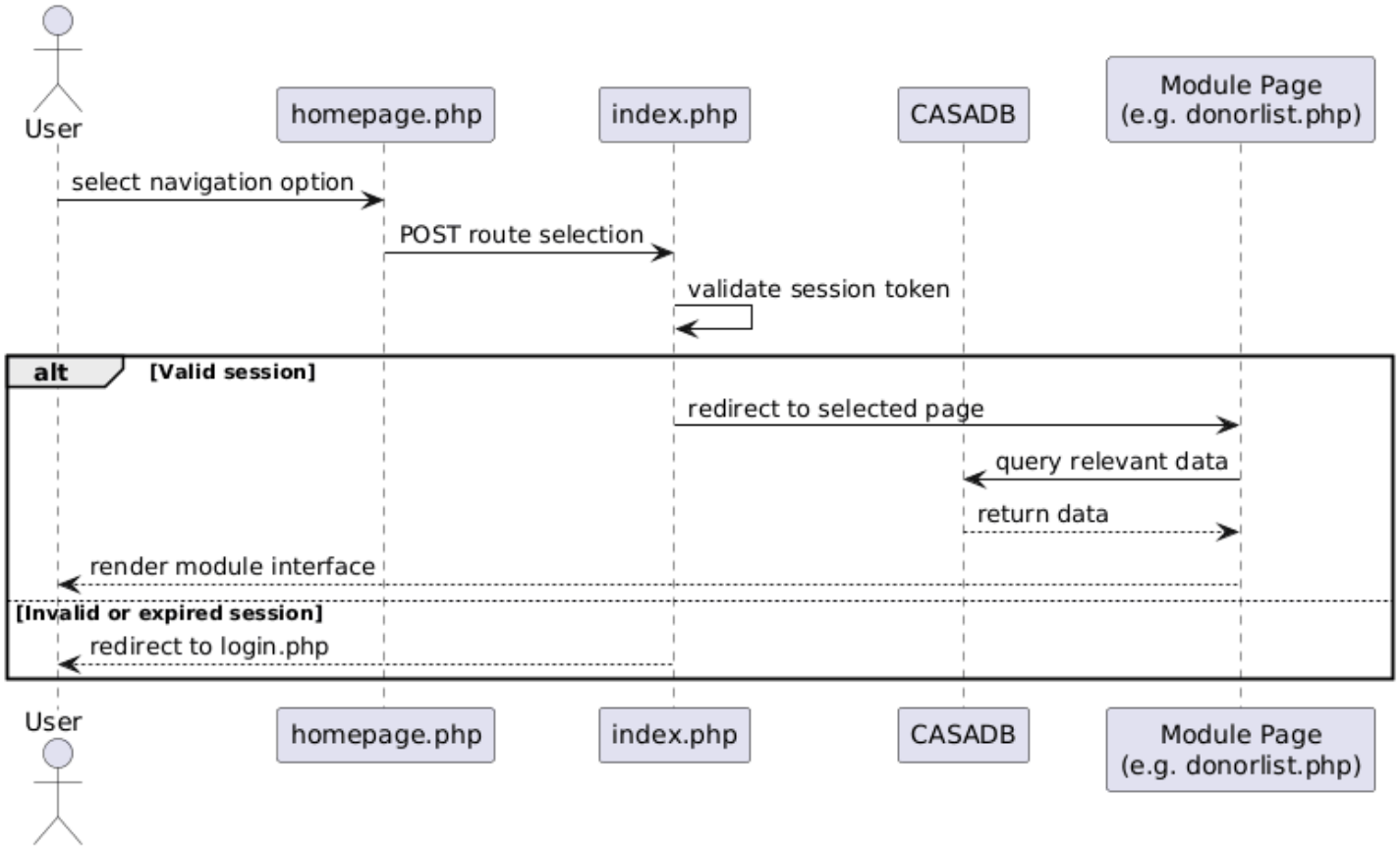
The exception flow handles cases where the connection to the API fails. If this occurs, the system returns an error message indicating that it could not connect.



4.1.4 Home Page Navigation

This sequence diagram illustrates the process of user navigation and session management within the system's homepage. After authentication, the user is directed to homepage.php, which serves as the primary navigation interface. When the user selects an option such as Donor Management, Event Calendar, or Reports, the page sends the selected route to index.php, which acts as the central routing controller. index.php validates the active session token and determines the appropriate destination. If the token is valid, the request is forwarded to the corresponding module (e.g., donorlist.php, calendar.php, or reportgen.php). The selected page then retrieves its associated data from CASADB through its backend handler (e.g., dbDonor.php or dbEvent.php), and the rendered page is displayed to the user.

In the event of an expired or invalid session, index.php redirects the user to login.php for re-authentication. This modular design enables consistent navigation, secure session handling, and scalability for future interface additions while minimizing redundant code across modules.



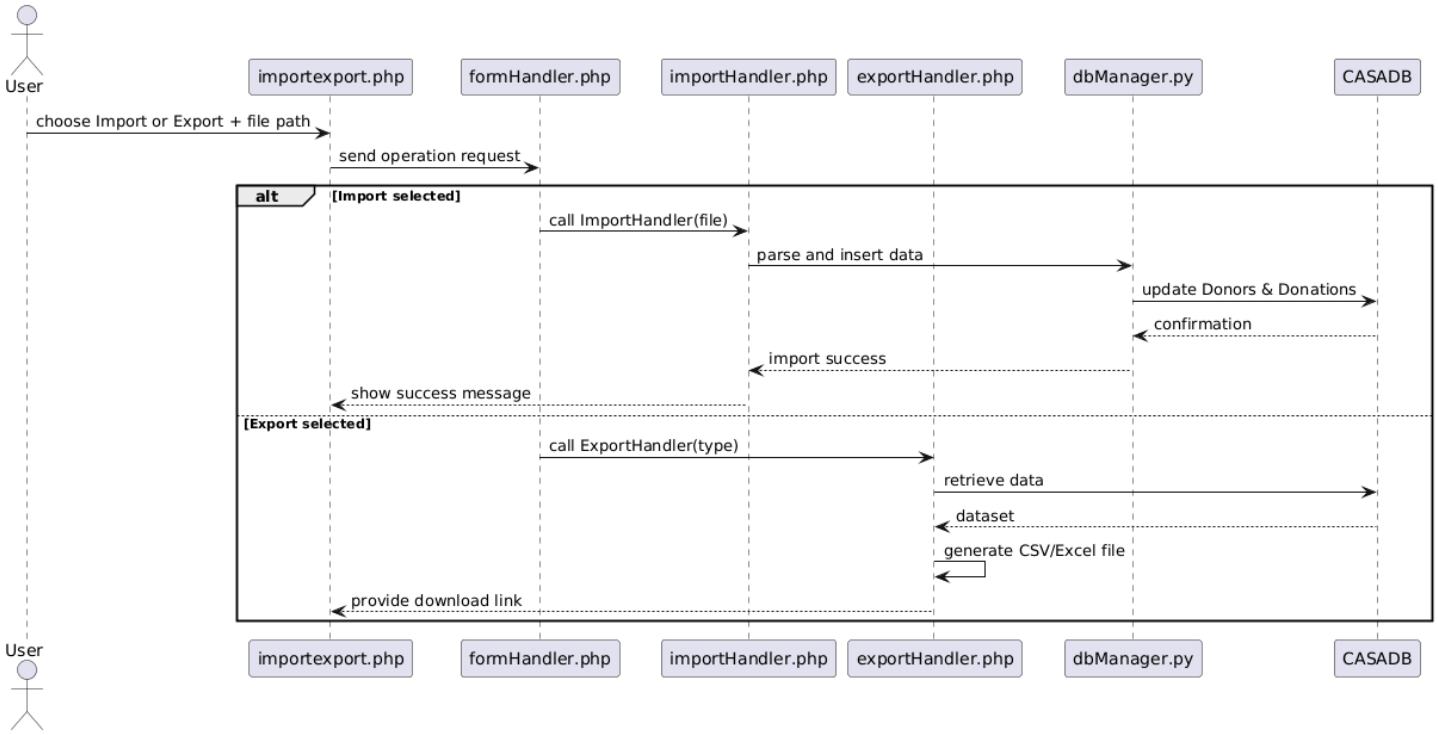
4.1.5 Import and Export

This sequence diagram illustrates how CASA staff import and export donor and donation data between the system and external files. The process begins when the user accesses importexport.php and selects either Import or Export. The

selection and file path are sent to formHandler.php, which routes the operation to importHandler.php or exportHandler.php depending on the chosen action.

During an import, importHandler.php validates the file type, then calls dbManager.py to parse and insert the data into the dbDonors and dbDonations tables in CASADB. Upon completion, the script returns a confirmation message to the web interface. For an export, exportHandler.php queries CASADB, compiles the retrieved dataset into an Excel or CSV file, and stores it in the specified directory. The resulting file path is then displayed to the user for download.

This process ensures reliable two-way data flow between CASA's system and external storage tools, simplifying recordkeeping, backups, and report distribution for staff members.



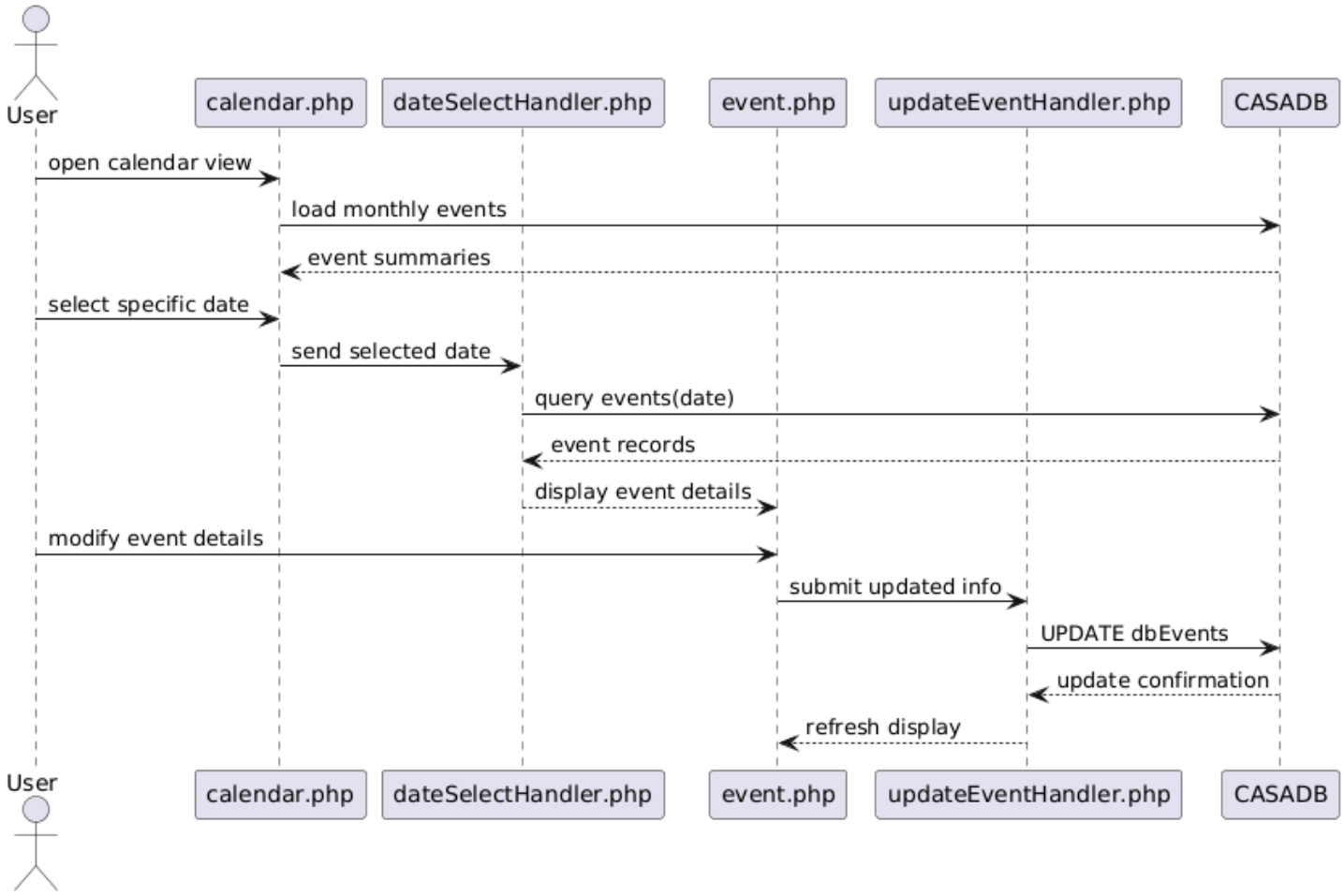
4.1.6 View Fundraising Calendar

This sequence diagram represents the workflow for viewing and editing fundraising events through the calendar interface. The user opens calendar.php, which loads the current month's events from CASADB. When the user selects a specific date, calendar.php sends that date to dateSelectHandler.php, which queries the *dbEvents* table for corresponding events.

The handler returns the retrieved event information to event.php, where the details are displayed in a structured form. If the user updates event data, the modified details are sent to updateEventHandler.php, validated, and written back to the *dbEvents* table. Upon successful update, the refreshed event data is returned to event.php, and the display updates accordingly.

This modular interaction supports seamless viewing and real-time editing of event records while maintaining accurate synchronization between the front-end calendar and the underlying database.

Note: Figure is displayed on the following page



4.1.7 Login

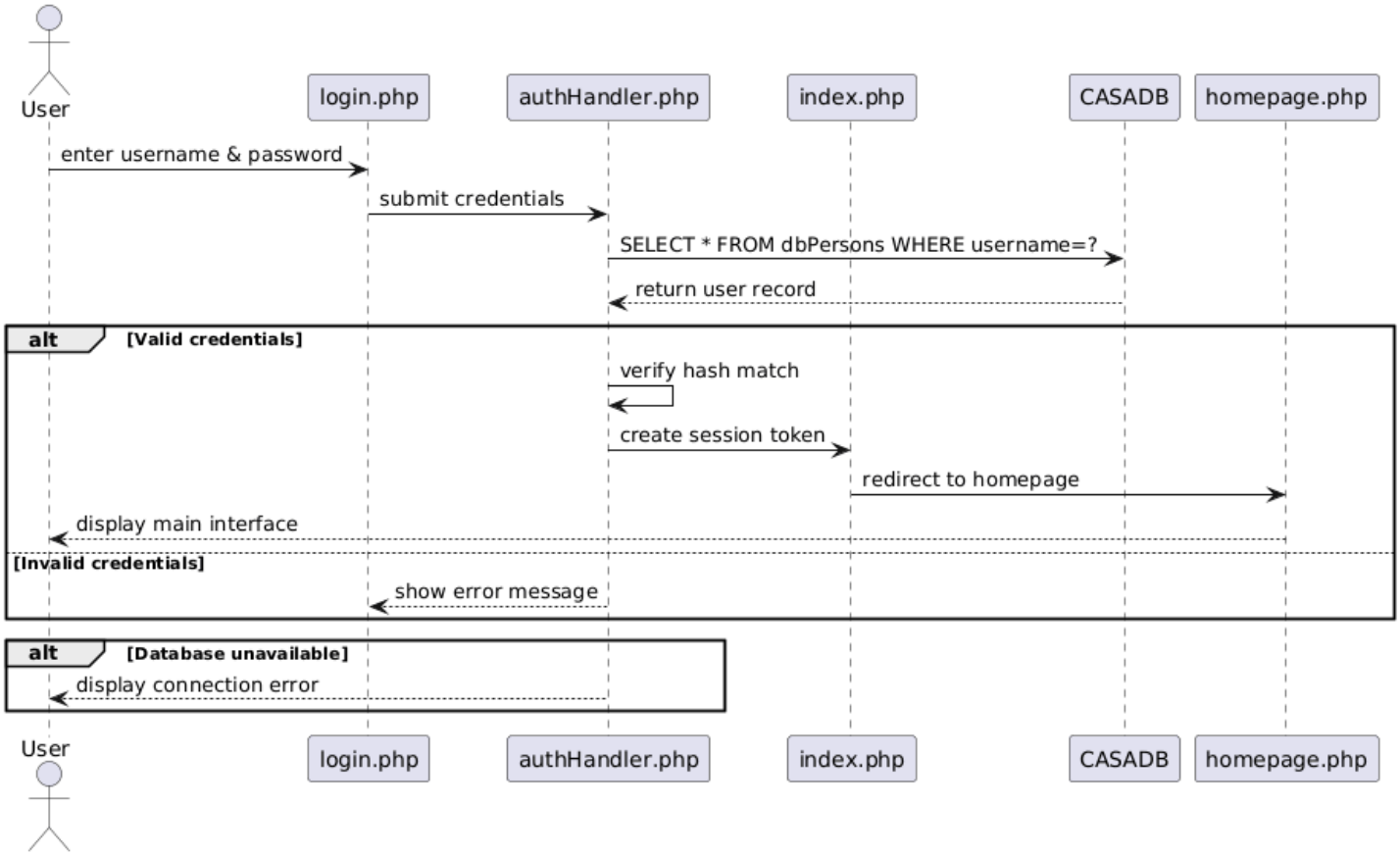
This sequence diagram outlines the login and authentication process for system users. When the user accesses login.php, they input their credentials and submit the form. The data is sent to authHandler.php, which hashes the password and queries the *dbPersons* table within CASADB for a matching username and hash.

If the credentials are valid, authHandler.php creates a session token and redirects the user to homepage.php. If the credentials are invalid, an error message is returned, prompting the user to retry.

An alternate flow occurs when a user attempts to access a restricted page without an active session. In such cases, index.php detects the missing session token and automatically redirects to login.php. Exception handling also covers cases where the database connection fails, displaying a notification banner and disabling further authentication attempts until the connection is restored.

This authentication sequence enforces access control, preserves data security, and maintains smooth user transitions between protected pages.

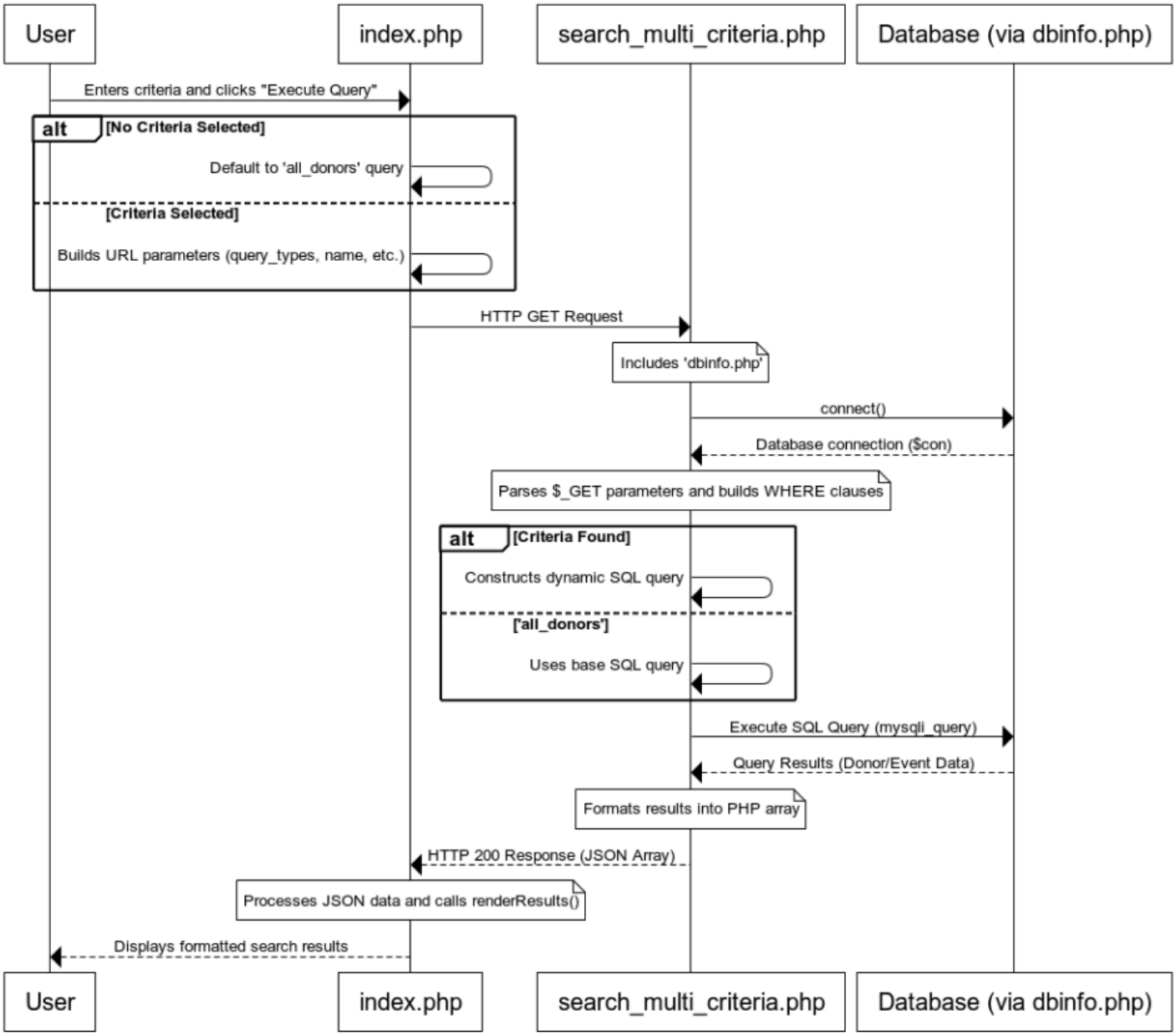
Note: Figure is displayed on the following page



4.1.8 Search

This sequence diagram outlines the donor search process for system users. The flow begins when the User submits search criteria from index.php. if there is no input default to all_donors query, if there is it will design an SQL search. This request is received by search_multi_criteria.php which first establishes a database connection via dbinfo.php. this then takes the clients parameters and builds an SQL Query. After generating the query, search_multi_criteria.php executes it against the Database. The Database returns the raw Query Results, which the PHP script processes, formats into an array. This data is then delivered back to index.php. The client-side JavaScript receives and processes this before presenting the formatted search results to the User without a page reload, ensuring a highly responsive and efficient user experience.

Note: Figure is displayed on the following page



5 Requirements Matrix

5.1 Matrix Description

This table describes the current state of our project. It contains the use cases and Scrum IDs which make up our backlog and their current completion status. Notably, some use cases have been extended through an additional sprint due to reworking required for the SiteGround transfer.

Requirement ID	Requirement Description	Associated Sprint	Status	Verification Method	Comments
UC 19	Automatically Generate Email	1-6	Reworking	Informal Testing On SiteGround	Needs to be reworked for SiteGround
Scrum-55	Home and Login Page Styling	1	Complete	N/A	Good stuff Max
UC 20	Send Emails	1-6	Reworking	Informal Testing On SiteGround	Needs to be reworked for SiteGround
UC 8	Export Donation/Donor Data	1-5	Complete	Informal Testing	Needed to be reworked for SiteGround but is now complete
UC 9	Import Donation/Donor Data	1-5	Complete	Informal Testing	Needed to be reworked for SiteGround but is now complete

Scrum-103	Build and Hydrate Database	1	Complete	N/A	Good stuff Carter
Scrum-104	Create New Users	1	Complete	N/A	Good stuff Garrett
UC 1	Add New Donor	7	Not Started	N/A	N/A
UC 2	Update Donor Info	7	Not Started	N/A	N/A
UC 3	Remove Donor	6	Not Started	N/A	N/A
UC 4	Update Donation Info	7	Not Started	N/A	N/A
UC 5	Manage Access Levels	N/A	Wishlisted	N/A	N/A
UC 6	Backup Donor Info	N/A	Wishlisted	N/A	N/A
UC 7	Add New Donation	7	Not Started	N/A	N/A

UC 10	Search Donors	6	Not Started	N/A	N/A
UC 11	Report Generation Form	2-3	Complete	Informal Testing	Good stuff Ethan
UC 12	Quantitative Report	2-5	In Progress	Demo Sent To Client	Needs more options for client
UC 13	Qualitative Report	2-5	In Progress	Demo Sent To Client	Needs more options for client
UC 14	Edit Fundraising Event	6	Not Started	N/A	N/A
UC 15	Create Fundraising Event	6	Not Started	N/A	N/A
UC 16	View Fundraising Events	6	Not Started	N/A	N/A
UC 17	Alert Emails	2	Complete	Peer Feedback	Good stuff Josh
UC 18	Thank You Emails	2	Complete	Peer Feedback	Good stuff Josh

Scrum-105	Style Email Page	3	Complete	Peer Feedback	Good stuff Max
Scrum-106	Style Import Page	4-5	Reworking	Peer Feedback	Change according to Polack's feedback
Scrum-107	Style Export Page	4-5	Reworking	Peer Feedback	Change according to Polack's feedback

6 Glossary

6.1 Definitions

API (Application Programming Interface): A set of protocols and tools that allows different software components to communicate and interact, such as the Hugging Face API used for email generation in this system.

CASADB: The central SQL database used by the CASA Donation Management Web Application to store information about donors, events, donations, emails, and reports.

DFD (Data Flow Diagram): A graphical representation that illustrates how data moves through a system, including inputs, processes, and outputs, as used in the high-level and low-level designs.

ERD (Entity-Relationship Diagram): A diagram that models the relationships between entities in a database, helping to visualize data structures like those in CASADB.

FK (Foreign Key): A field in a database table that links to the primary key in another table, enabling relationships between data entities such as donors and donations.

JSON (JavaScript Object Notation): A lightweight data interchange format used for structuring data, often in API responses like those from the Hugging Face API.

Matplotlib: A Python library for creating static, animated, and interactive visualizations, used in the report generation process for graphs and charts.

Normalization: A database design technique that reduces redundancy and improves data integrity by organizing data into related tables.

Ollama Container: The local AI generation environment used for creating and testing automated email content.

Pandas: A Python library for data manipulation and analysis, employed in the backend to process and query data from CASADB for reports.

PHP (Hypertext Preprocessor): A server-side scripting language used for web development, forming the basis of many system components like login.php and formHandler.php.

PK (Primary Key): A unique identifier for each record in a database table, ensuring data integrity and uniqueness, such as the 'id' field in dbEvents.

Python: A versatile programming language used for backend scripting in this application, including data analysis with libraries like pandas and seaborn.

Report Generation: The automated process of analyzing stored data to create PDF or visual reports for CASA staff.

Seaborn: A Python data visualization library built on matplotlib, utilized for creating informative statistical graphics in reports.

Sequence Diagram: A type of UML diagram that shows interactions between objects over time, detailing workflows like event creation or login processes.

SQL (Structured Query Language): The standard language for managing and querying relational databases, used to interact with CASADB for storing and retrieving data.

7 Author Information

7.1 Author Emails

Maximilian Redman: mredman@mail.umw.edu

Carter Walker: cwalke28@mail.umw.edu

James Heathcock: jheathco@mail.umw.edu

Joshua Byrne: jbyrne@mail.umw.edu

Garrett McKenzie: gmckenzi@mail.umw.edu

Ethan Bostick: ebostick@mail.umw.edu

7.2 Author Contributions

Joshua Byrne - I created the sequence diagrams and descriptions for Sections 4.1.4–4.1.7 (Home Page Navigation, Import & Export, View Fundraising Calendar, and Login). After building those, I went back and standardized the earlier diagrams in 4.1.1–4.1.3 so all seven share the same style. I also matched the write-ups to the diagrams so the flows line up with the actual PHP/Python handlers and database calls. In short: I created the new diagrams (4.1.4–4.1.7) and then made sure every diagram in 4.1.1–4.1.7 look and read consistently for the final paper.

Ethan Bostick - I authored the Statement of Purpose, Intended Audience, Scope, References, Overview, and both the High and Low-Level Design sections (1.1–2.2). Additionally, I wrote the process descriptions for Email Generation, Homepage Navigation, Report Generation, and Fundraising Event Creation (2.2.1–2.2.3, 2.2.5). I created the Data Flow Diagrams (DFDs) for the Report Generation, Homepage Navigation, and Fundraising Event Creation processes. Furthermore, I completed Sections 3.1 and 3.2, and authored the introductory description for the Sequence Diagrams, along with the detailed descriptions for the Create Fundraising Event, Report Generation, and Email Generation Sequence Diagrams. I also created the Sequence Diagram for the Create Fundraising Event process.

Garrett McKenzie - I authored the data-flow diagrams and descriptions for the import/export process as well as the fundraising calendar process (4.1.5 and 4.1.6). Additionally, I assisted Ethan with the creation of the high-level data flow diagram (2.1) and the database design found throughout section 3. Finally, I worked on the requirements matrix and requirements matrix description found in section 5.

Carter Walker - I was responsible for authoring the data-flow diagrams, sequence diagrams, and descriptions for the Login process and search (2.2.7 and 4.1.8). Additionally I helped Ethan and Garret standardize wording and making minor edits in sections (2 and 4). I was responsible for making numerous minor edits and corrections to enhance clarity and accuracy. In conclusion I mostly helped with editing and finalizing the paper aside from the specific sections that I was assigned.

Max Redman - I authored most of the definitions. I also edited the document for consistency by correcting capitalization, aligning section numbering, and indentation across all levels, and ensuring that all embedded diagrams and figures are properly referenced and displayed on the following pages as noted. I also changed the data dictionary to reflect exactly what the ERD says.