Software Requirements Specification

Uvents

25 June 2024

Scope

Uvents is a mobile application that brings the community together by allowing

- consumer users to create or participate in events, which can be set as a group event or a Date event
- commercial users to list their promotions as an event, which can be seen by the personal users and added to their events
- admin users to manage general event policies, monitor user activity, and maintain the application

This document includes the design representation and the design information for the application components, classes, function, databases, and data stores. It describes how data is represented in the system, as well as the algorithms used by the application systems.

Stakeholders

Sponsor: James Helfrich

Project Manager: Annmarie SanSevero

Vice-Project Manager: James Green

Chief of Technology: Reagan Houser

Chief of Editors: Brienna Sambrano

Chief of QA: Fabian Diaz Santiago

Contributors

Aaron Fisk
Aaron Fox
Alain Catari
Ammon Jones
Annmarie SanSevero
Aria Menke
Arunas Rancevas
Ben Wassum
Benjamin Smith
Brienna Sambrano
Brighton Hatch
Brian D Crews
Calvin Bullock
Cal Quinton
Craig Conover
Craig Mortensen
Dalan Weber
David Raya Lucero
Eros Pereira
Emory Hubbard
Evan Handy
Fabian Diaz Santiago
Garren Diab
Gavin Cannon
Godspower Okonkwo
Isaac Radford
James DeMordaunt
James Green
Jennifer Weber
Jovani Ambriz
Josh Liddiard
Joshua Johnson
Joshua Owolabi Olaoye
Jorge Alberto Chavez Ponce
Joseph Gregory
Joseph Santos
John Lydiksen
JoE Nnachi
Kai Zhang
Kendra Bryant
Matthew Thomas Knorr
Michael Darling
Miguel Angel Lopez Guzman
Nathan Spotten
Pume Cartagena
Rayshorn Richardson
Reagan Houser
Richard Gibbons III
Ryan Donaldson
Sarah Steadman
Taemour Djahanbani
Tanner Dillon
Victor Elerunndu

Contents

Change History

Draft	Changes Made	Date
1	Added in the first component diagram	11 May 2024
2	Added additional views, front matter	28 May 2024
3	Added additional component views, back matter	11 June 2024
4	Revised views, added class diagrams and database tables; added links within document	25 June 2024

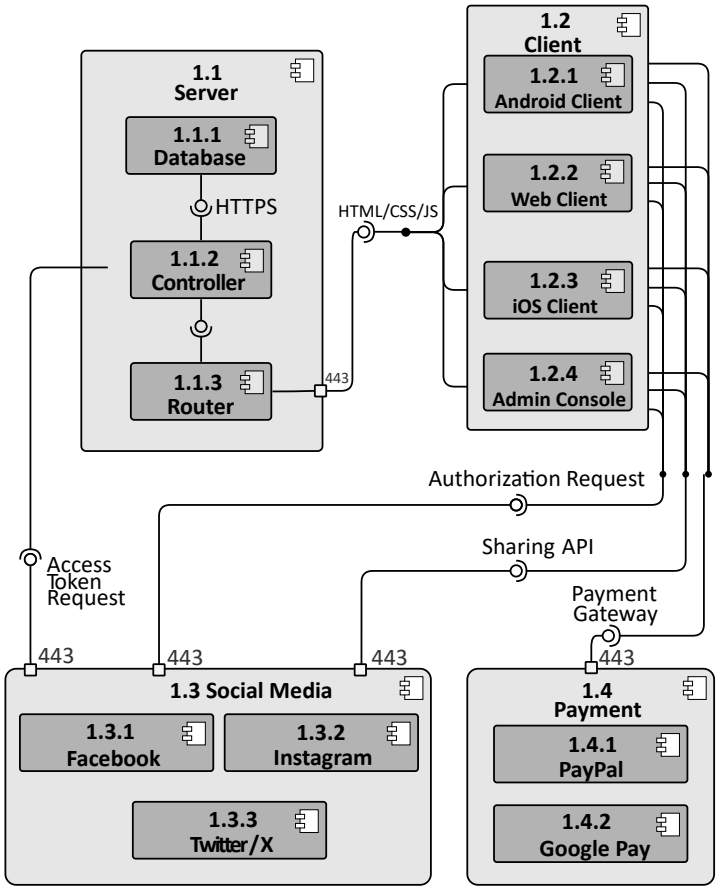
References

Source	Description
Uvents SRS	Software requirement specification for the Uvents application

Viewpoints

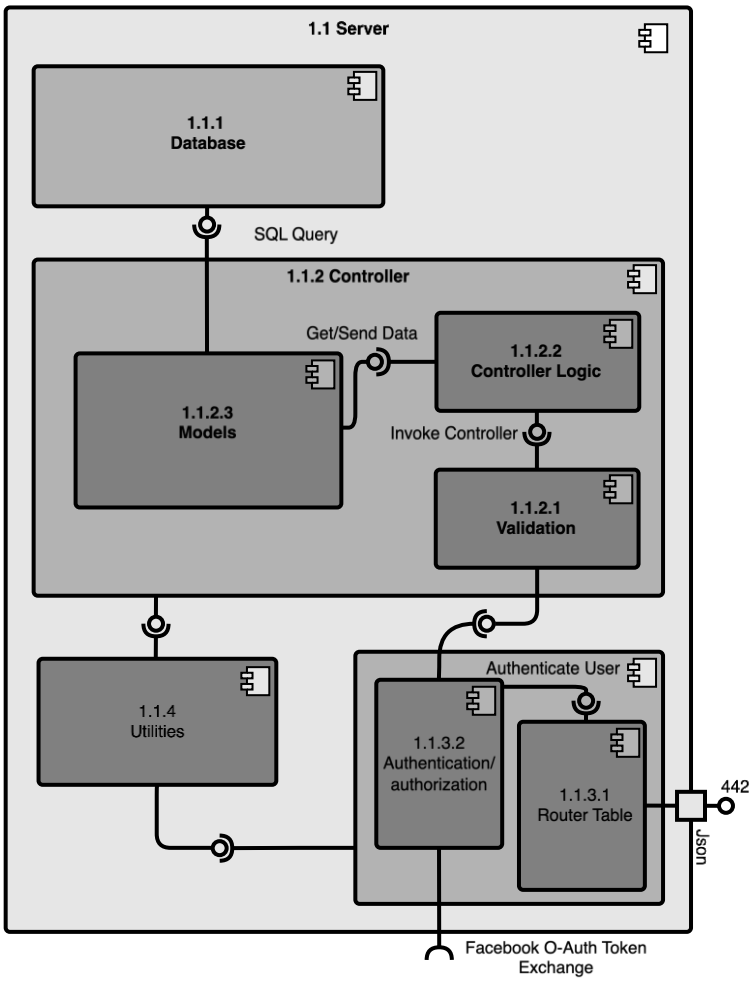
Viewpoint	Source
Component Diagram	IBM Developer. "The Component Diagram." [Online]. Available: https://developer.ibm.com/articles/the-component-diagram/ [Accessed: May 27, 2024].
Structure Chart	GeeksforGeeks. "Software Engineering | Structure Charts." [Online]. Available: https://www.geeksforgeeks.org/software-engineering-structure-charts/ [Accessed: May 27, 2024].
Data Flow Diagram	Visual Paradigm. "What is Data Flow Diagram (DFD)?" [Online]. Available: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/ [Accessed: May 27, 2024].
Wireframe	Balsamiq. "What are Wireframes?" [Online]. Available: https://balsamiq.com/learn/articles/what-are-wireframes/ [Accessed: May 27, 2024].
Workflow Diagram	Lucidchart. "Workflow Diagram Tutorial." [Online]. Available: https://www.lucidchart.com/pages/tutorial/workflow-diagram [Accessed: May 27, 2024].
Sequence Diagram	GeeksforGeeks. "Unified Modeling Language (UML) | Sequence Diagrams." [Online]. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/ [Accessed: May 27, 2024].
Entity Relationship Diagram	Lucidchart. "ER Diagrams." [Online]. Available: https://www.lucidchart.com/pages/er-diagrams [Accessed: May 27, 2024].
Site Map	S. Enlow, "What Is a Sitemap? 4 Sitemap Examples & Best Practices," Markitors, Sep. 20, 2022. [Online]. Available: https://markitors.com/what-is-a-sitemap/. [Accessed: Jun. 7, 2024].
Class Diagram	GeeksforGeeks, "Unified Modeling Language (UML) | Class Diagrams," GeeksforGeeks. Available: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/. [Accessed: June 10, 2024].
Flow Chart	GeeksforGeeks, "An Introduction to Flowcharts," GeeksforGeeks. Available: https://www.geeksforgeeks.org/an-introduction-to-flowcharts/. [Accessed: June 11, 2024].
Pseudocode	"How to Write a Pseudo-code." GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/. [Accessed: 24-Jun-2024].
JSON Schema	"JSON Schema: Specification." JSON Schema. [Online]. Available: https://json-schema.org/specification. [Accessed: 24-Jun-2024].

View 1: Uvents Component Diagram



Name	1 Uvents Component Diagram	
Purpose	Describe overall system view	
Description	Illustrate the main components and connections between components for the Uvents application	
Requirements	PO 1.1-PO 4.9	
Elements	**1.1 Server**	**1.2.4 Admin Console**
	1.1.1 Database	**1.3 Social Media**
	1.1.2 Controller	**1.3.1 Facebook**
	1.1.3 Router	**1.3.2 Instagram**
	1.2 Client	**1.3.3 Twitter/X**
	1.2.1 Android Client	**1.4 Payment**
	1.2.2 Web Client	**1.4.1 PayPal**
	1.2.3 iOS Client	**1.4.2. Google Pay**
Referenced By		
Viewpoint	Component Diagram	

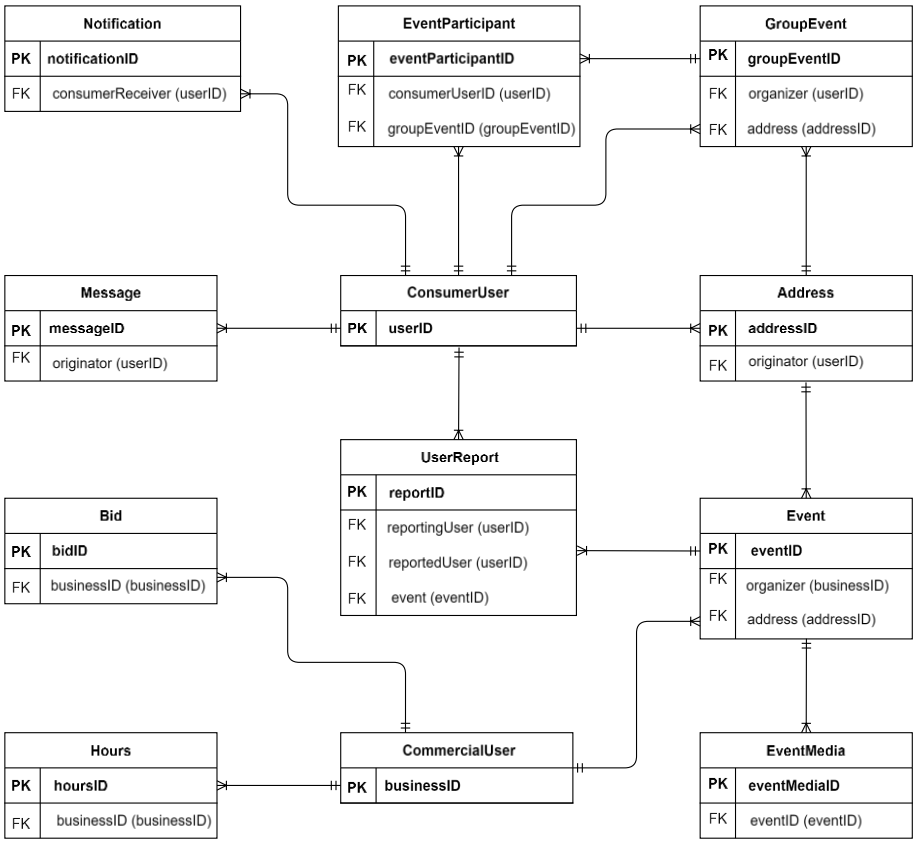
View 1.1: Server



Name	1.1: Server	
Purpose	Provides a high-level view on the app server architecture.	
Description	Displays the app server components and their connections. Follows typical Express architecture.	
Requirements	PO.1.1.3, SAV.2.2, SAV.2.5.2, SAV.4.2	
Elements	1.1.1 **Database:** Relational database for the application server	**Controller-Models:** Calls various model CRUD operations
	1.1.3.1 **Router Table:** Module handling all API server routes	1.1.2.2 **Controller:** Applies logic to the request and supplied JSON data
	Authentication/Authorization: Beginning of the middleware chain	1.1.2.3 **Models:** Model components that interface with the database
	1.1.3.2 **Authentication/Authorization:** Module that authenticates users. Can use the Facebook Oauth API.	1.1.4 **Utilities:** A collection of useful, general-purpose functions

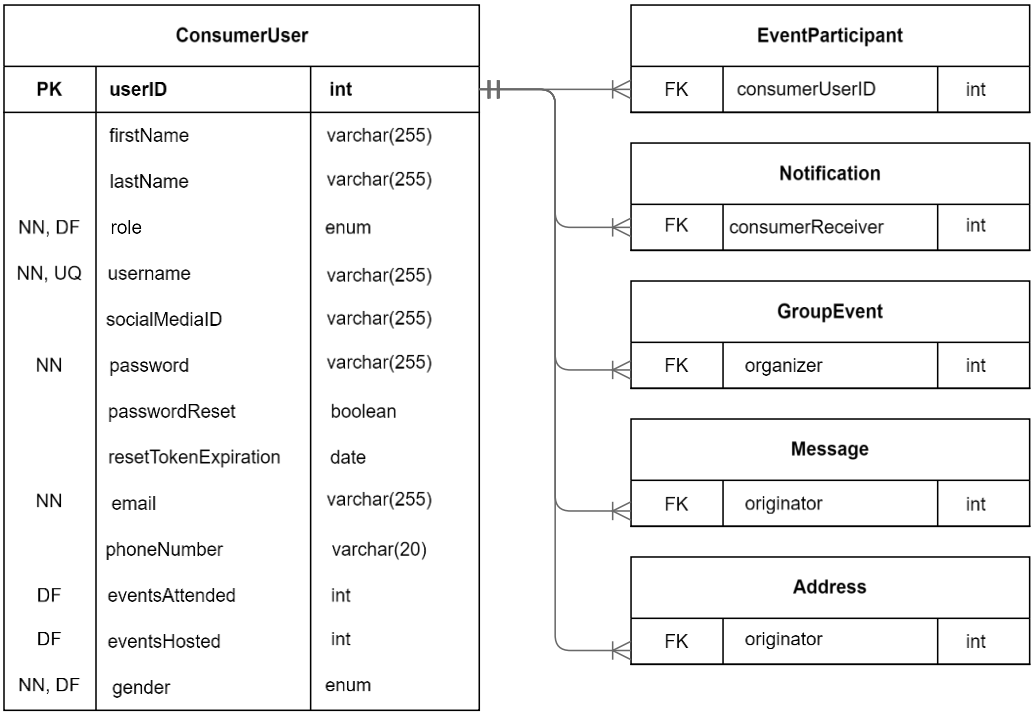
	Authentication/Authorization-Validation: Express middleware next() call	**1.1.5 Endpoints:** A listing of all API endpoints.
	1.1.2.1 Validation: Module that validates client JSON data	**Router-Validation-Controller:** The end of the middleware stack
Referenced by	1	
Viewpoint	Component Diagram	

View 1.1.1: Database Overview



Name	1.1.1 DatabaseOverview
Purpose	Database structure for the Uvents application
Description	Shows relationship of all database tables foreign keys
Requirements	3.5
Elements	1.1.1.11 Notification
	1.1.1.6 EventParticipant
	1.1.1.5 GroupEvent
	1.1.1.10 Message
	1.1.1.1 ConsumerUser
	1.1.1.13 Address
	1.1.1.12 Bid
	1.1.1.7 UserReport
	1.1.1.3 Event
	1.1.1.14 Hours
	1.1.1.2 CommercialUser
	1.1.1.4 EventMedia
Referenced By	View 1: Uvents Component Diagram #1
Viewpoint	Entity Relationship Diagram

View 1.1.1.1 ConsumerUser



Name	**1.1.1.1 ConsumerUser**
Purpose	Describe the Consumer User Table fields, types, and constraints, and its relationship to other tables.
Description	The Consumer User Table will store all the information needed to represent a given consumer user.
Requirements	3.5.1
Elements	**role**: DF = consumerUser, commercialUser, or adminUser
	username: unique identifier for signing in
	socialMediaID: which social media platform they are signing in with
	passwordReset: true when a password needs a reset
	resetTokenExpiration: expiration date of password
	eventsAttended: DF = 0
	eventsHosted: DF = 0
	gender: male, female, or DF = unspecified
	1.1.1.6 EventParticipant
	1.1.1.11 Notification
	1.1.1.5 GroupEvent
	1.1.1.10 Message
	1.1.1.13 Address
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.1.1 ConsumerUser SQL

Pseudocode

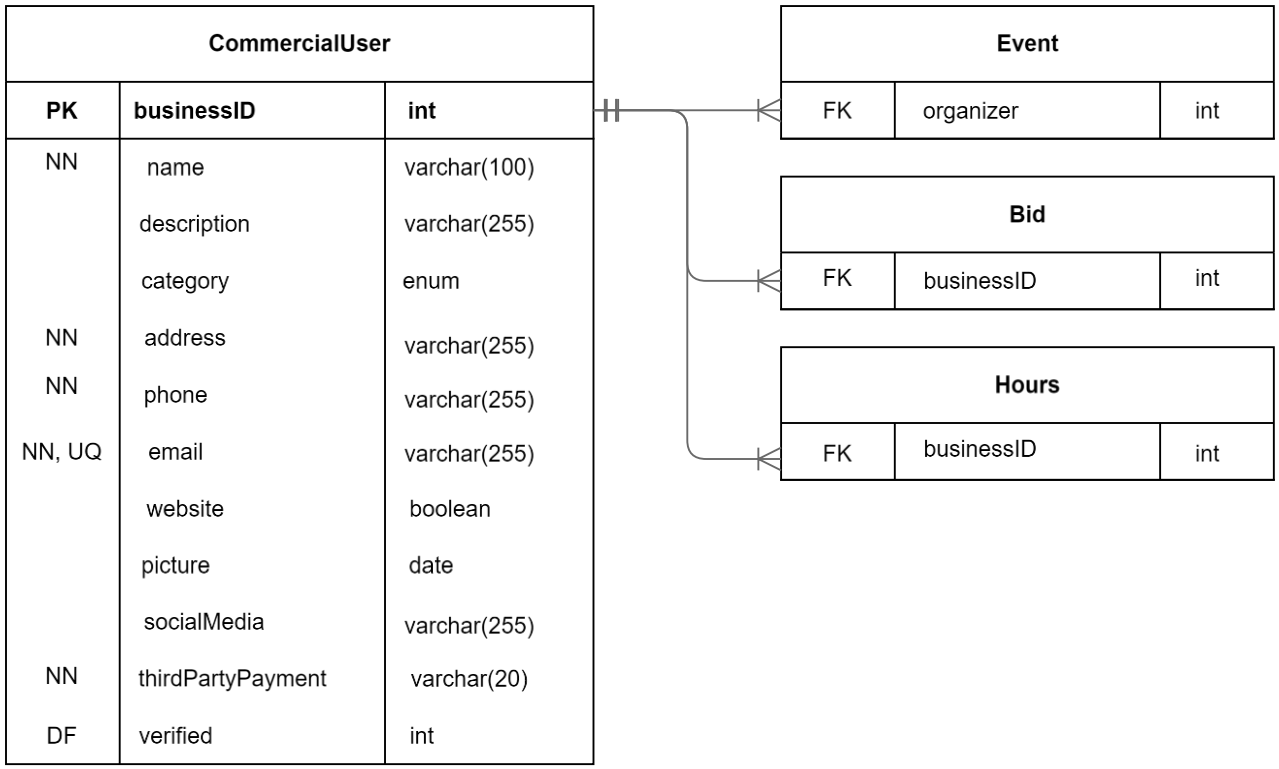
```
CREATE TYPE role_enum AS ENUM (
   'consumerUser', 'commercialUser', 'adminUser'
);

CREATE TYPE gender_enum AS ENUM (
   'male, 'female', 'unspecified'
);

CREATE TABLE consumerUser (
   userID serial4 NOT NULL,
   firstName varchar(100),
   lastName varchar(100),
   role role_enum NOT NULL DEFUALT “consumerUser”,
   username varchar(255) NOT NULL UNIQUE,
   socialMediaID varchar(255),
   password varchar(255) NOT NULL,
   passwordReset BOOLEAN DEFAULT FALSE,
   resetTokenExpiration DATE,
   email varchar(255) NOT NULL,
   phone varchar(20),
   eventsAttended int DEFAULT 0,
   eventsHosted int DEFAULT 0,
   gender gender_enum DEFAULT “unspecified”,
   CONSTRAINT groupEventID_pkey PRIMARY KEY (groupEventID),
   CONSTRAINT fk_organizerID FOREIGN KEY (organizerID) REFERENCES consumerUser (userID),
   CONSTRAINT fk_addressID FOREIGN KEY (addressID) REFERENCES address (addressID)
);
```

Name	**1.1.1.1.1 ConsumerUser SQL**
Purpose	Describe the Consumer User Table fields, types, and constraints, and its relationship to other tables.
Description	The Consumer User Table will store all the information needed to represent a given consumer user.
Requirements	3.5.1
Elements	**username**: unique identifier for signing in
	socialMediaID: which social media platform they are signing in with
	passwordReset: true when a password needs a reset
	resetTokenExpiration: expiration date of password
Referenced By	1.1.1.1
Viewpoint	Pseudocode

View 1.1.1.2 CommercialUser



Name	1.1.1.2 CommercialUser
Purpose	Describe the CommercialUser fields, types, and constraints and its relationship to other tables
Description	The CommercialUser will store all the information needed to represent a given Commercial User
Requirements	3.5.2
Elements	**category**: predetermined options (FS.15.4.3) - no options given
	socialMedia: link to businesses social media page
	thirdPartyPayment: stores payment information
	verified: true if business has been verified by an admin, DF = false
	1.1.1.12 Bid
	1.1.1.14 Hours
	1.1.1.3 Event
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.2.1 CommercialUser SQL

Pseudocode

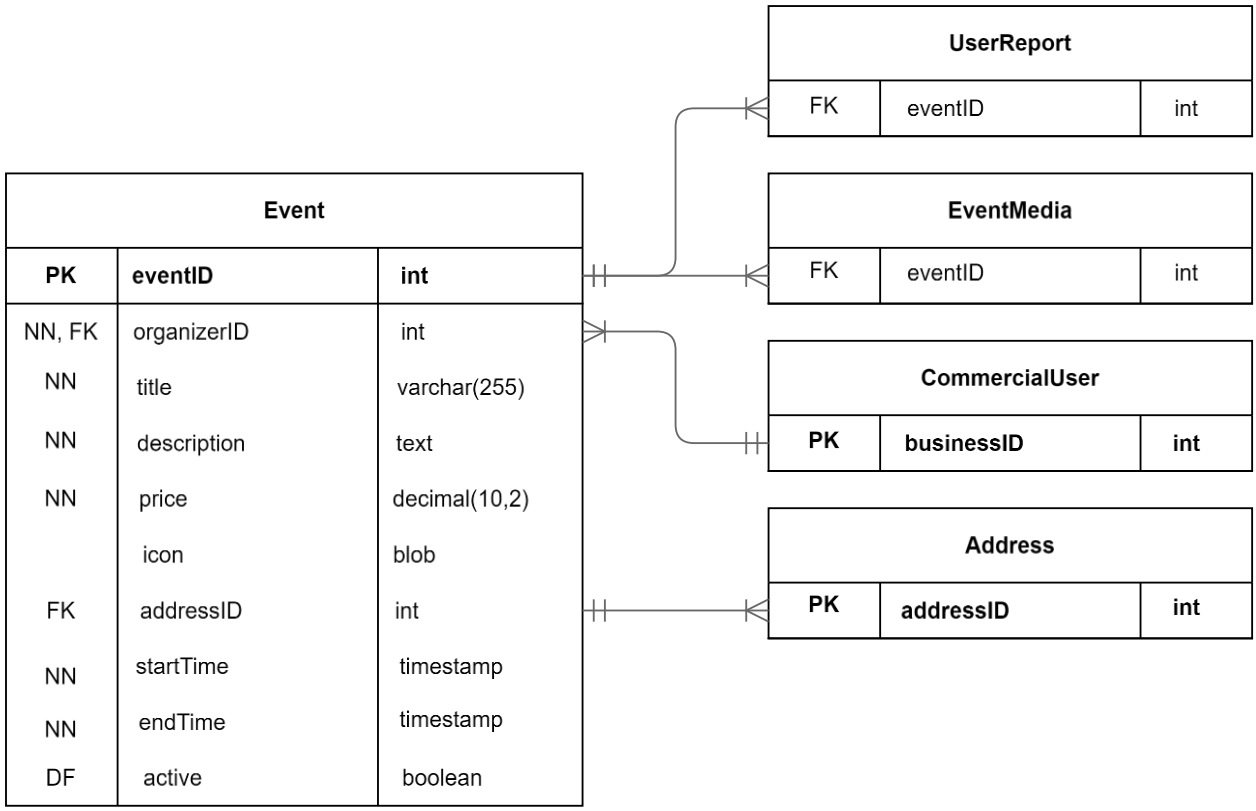
```
CREATE TYPE category_enum AS ENUM (

);

CREATE TABLE commercialUser (
   businessID serial4 NOT NULL,
   name varchar(100) NOT NULL,
   description varchar(255),
   category category_enum,
   address varchar(255) NOT NULL,
   phone varchar(100) NOT NULL,
   email varchar(255) NOT NULL UNIQUE,
   website varchar(255),
   picture blob,
   socialMedia varchar(255),
   thirdPartyPayment varchar(20) NOT NULL,
   verified BOOLEAN DEFAULT FALSE,
   CONSTRAINT businessID_pkey PRIMARY KEY (businessID),
);
```

Name	**1.1.1.2.1 CommercialUser SQL**
Purpose	Provide the SQL for generating the CommercialUser table
Description	The CommercialUser SQL details all fields, types, and constraints that make up the ConsumerUser table.
Requirements	3.5.2
Elements	**category_enum**: predetermined options (FS.15.4.3). No options given
	socialMedia: link to businesses social media page
	thirdPartyPayment: stores payment information
	verified: true if business has been verified by an admin
Referenced By	1.1.1.2
Viewpoint	Pseudocode

View 1.1.1.3 Event



Name	1.1.1.3 Event
Purpose	Describe the Events fields, types, and constraints and its relationship to other tables
Description	Event will store all the information needed to represent a given Event created by a business.
Requirements	SRS 3.5.3
Elements	**title**: Event name
	icon: optional image for the Event
	active: indicates if the event is currently ongoing, DF = false
	1.1.1.2 CommercialUser
	1.1.1.13 Address
	1.1.1.7 UserReport
	1.1.1.4 EventMedia
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

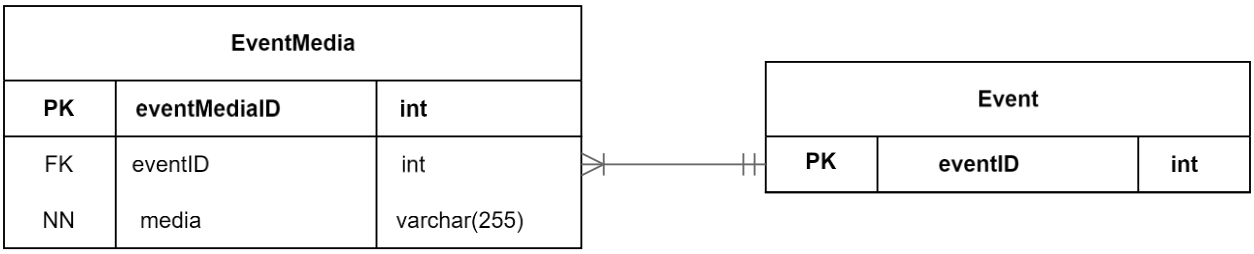
View 1.1.1.3.1 Event SQL

Pseudocode

```
CREATE TABLE event (
    eventID serial4 NOT NULL,
    organizerID int4 NOT NULL,
    title varchar(255) NOT NULL,
    description varchar(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    icon blob,
    addressID int4 NOT NULL,
    startTime DATETIME NOT NULL,
    endTime DATETIME NOT NULL,
    active BOOLEAN DEFAULT FALSE,
    CONSTRAINT eventID_pkey PRIMARY KEY (eventID),
    CONSTRAINT fk_organizerID FOREIGN KEY (organizerID) REFERENCES consumerUser (userID),
    CONSTRAINT fk_addressID FOREIGN KEY (addressID) REFERENCES address (addressID)
);
```

Name	**1.1.1.3.1 Event SQL**
Purpose	Provide the SQL for generating the Event table.
Description	The Event SQL details all fields, types and constraints that make up the Event table.
Requirements	SRS 3.5.3
Elements	**title**: Event name
	icon: optional image for the Event
	active: indicates if the event is currently ongoing
Referenced By	1.1.1.3
Viewpoint	Pseudocode

View 1.1.1.4 EventMedia



Name	1.1.1.4 EventMedia
Purpose	Describes EventMedia fields, types, constraints, and its relationship to other tables.
Description	EventMedia will store all the information needed to represent media associated with an event
Requirements	3.5.4
Elements	**media**: represents a media item (LDR.4) which a Commercial User can upload (FS.19.8)
	1.1.1.3 Event
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

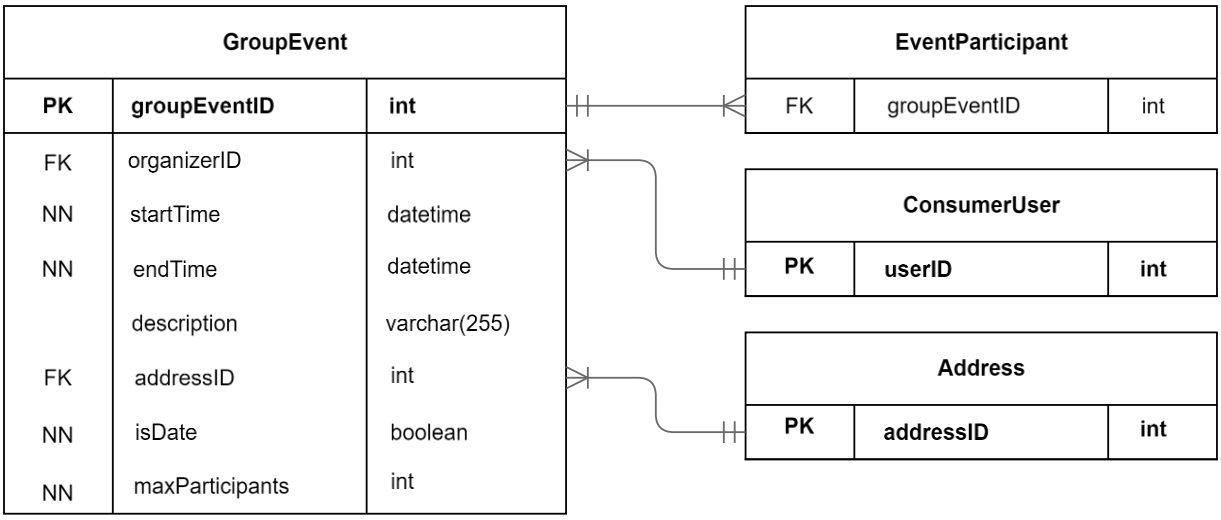
View 1.1.1.4.1 EventMedia SQL

Pseudocode

```
CREATE TABLE eventMedia  (
   eventMediaID serial4 NOT NULL,
   eventID int4 NOT NULL,
   media varchar(255) NOT NULL,
   CONSTRAINT eventMediaID_pkey PRIMARY KEY (eventMediaID),
   CONSTRAINT fk_eventID FOREIGN KEY (eventID) REFERENCES event (eventID)
);
```

Name	**1.1.1.4.1 EventMedia SQL**
Purpose	Describes EventMedia fields, types, constraints, and its relationship to other tables.
Description	EventMedia will store all the information needed to represent media associated with an event
Requirements	3.5.4
Elements	**media**: represents a media item (LDR.4) which a Commercial User can upload (FS.19.8)
Referenced By	1.1.1.4
Viewpoint	Pseudocode

View 1.1.1.5 GroupEvent



Name	1.1.1.5 GroupEvent
Purpose	Describe the group event table fields, types, and constraints and its relationship to other tables
Description	The Group Event Table will store all the information needed to represent a group event
Requirements	3.5.5
Elements	**isDate**: whether an event is intended to be a Date
	1.1.1.1 ConsumerUser
	1.1.1.13 Address
	1.1.1.6 EventParticipant
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

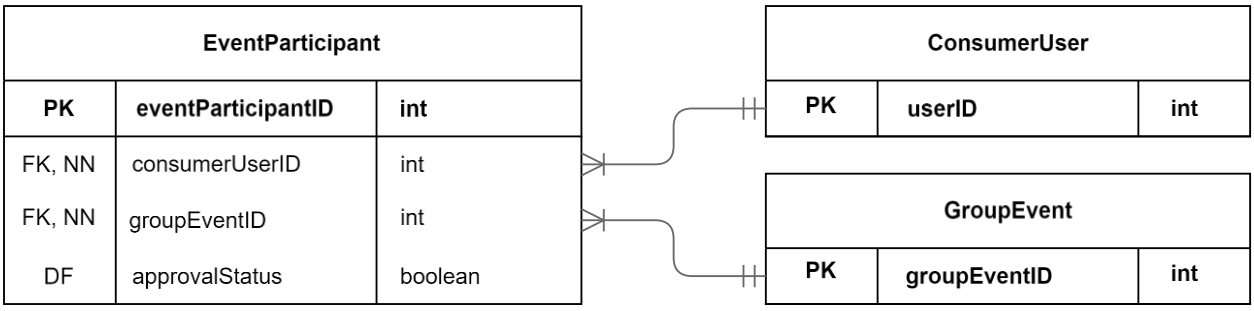
View 1.1.1.5.1 GroupEvent SQL

Pseudocode

```
CREATE TABLE eventMedia  (
   groupEventID serial4 NOT NULL,
   organizerID int4 NOT NULL,
   startTime DATETIME NOT NULL,
   endTIME DATETIME NOT NULL,
   description varchar(255),
   addressID int4,
   isDate BOOLEAN,
   maxParticipants int NOT NULL,
   CONSTRAINT groupEventID_pkey PRIMARY KEY (groupEventID),
   CONSTRAINT fk_organizerID FOREIGN KEY (organizerID) REFERENCES consumerUser (userID)
   CONSTRAINT fk_addressID FOREIGN KEY (addressID) REFERENCES address (addressID)
);
```

Name	**1.1.1.5.1 GroupEvent SQL**
Purpose	Describe the group event table fields, types, and constraints and its relationship to other tables
Description	The Group Event Table will store all the information needed to represent a group event
Requirements	3.5.5
Elements	**isDate**: whether an event is intended to be a Date
Referenced By	1.1.1.5
Viewpoint	Pseudocode

View 1.1.1.6 EventParticipant



Name	1.1.1.6 EventParticipant
Purpose	Describe the Event Participants fields, types, and constraints and its relationship to other tables
Description	The Event Participants Table will store all the information needed to represent a participant attending an event.
Requirements	3.5.6
Elements	**approvalStatus**: informs the user of approval for the event, default = false
	1.1.1.1 ConsumerUser
	1.1.1.5 GroupEvent
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

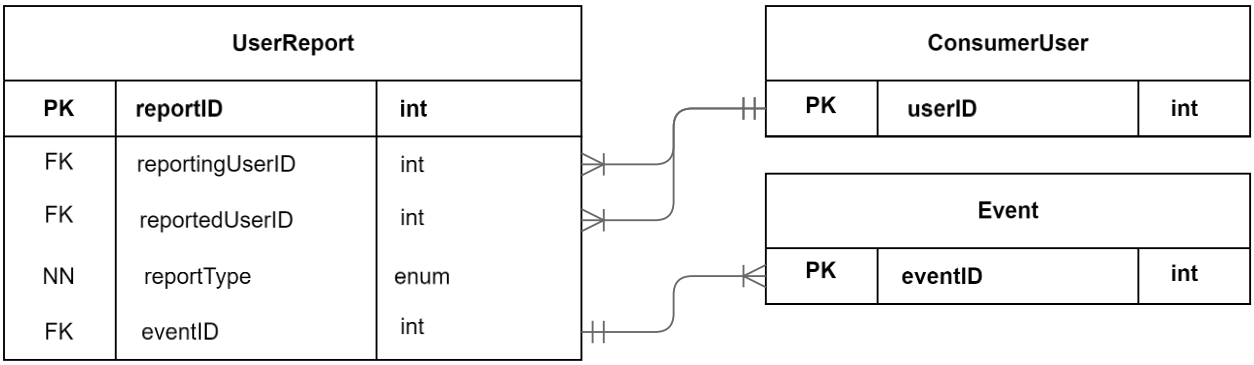
View 1.1.1.6.1 EventParticipant SQL

Pseudocode

```
CREATE TABLE eventParticipant (
    eventParticipantID serial4 NOT NULL,
    consumerUserID int4 NOT NULL,
    groupEventID int4 NOT NULL,
    approvalStatus BOOLEAN DEFAULT FALSE,
    CONSTRAINT eventParticipantID_pkey PRIMARY KEY (eventParticipantID),
    CONSTRAINT fk_consumerUserID FOREIGN KEY (consumerUserID) REFERENCES consumerUser (userID),
    CONSTRAINT fk_groupEventID FOREIGN KEY (groupEventID) REFERENCES groupEvent (groupEventID)
);
```

Name	**1.1.1.6.1 EventParticipant SQL**
Purpose	Describe the Event Participants fields, types, and constraints and its relationship to other tables
Description	The Event Participants Table will store all the information needed to represent a participant attending an event.
Requirements	3.5.6
Elements	**approvalStatus**: informs the user of approval for the event, default = false
Referenced By	1.1.1.6
Viewpoint	Pseudocode

View 1.1.1.7 UserReport



Name	1.1.1.7 UserReport
Purpose	Shows a relationship between a reported user, a report type, and an associated event (if applicable) for an admin to review.
Description	The UserReport table will store all the information regarding reports.
Requirements	3.5.7
Elements	**reportType**: options: “No Show” (LDR 7.4.1), “Spam”, or “Fake Account” (SAS.3, SAV.3).
	1.1.1.1 ConsumerUser
	1.1.1.3 Event
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.7.1 UserReport SQL

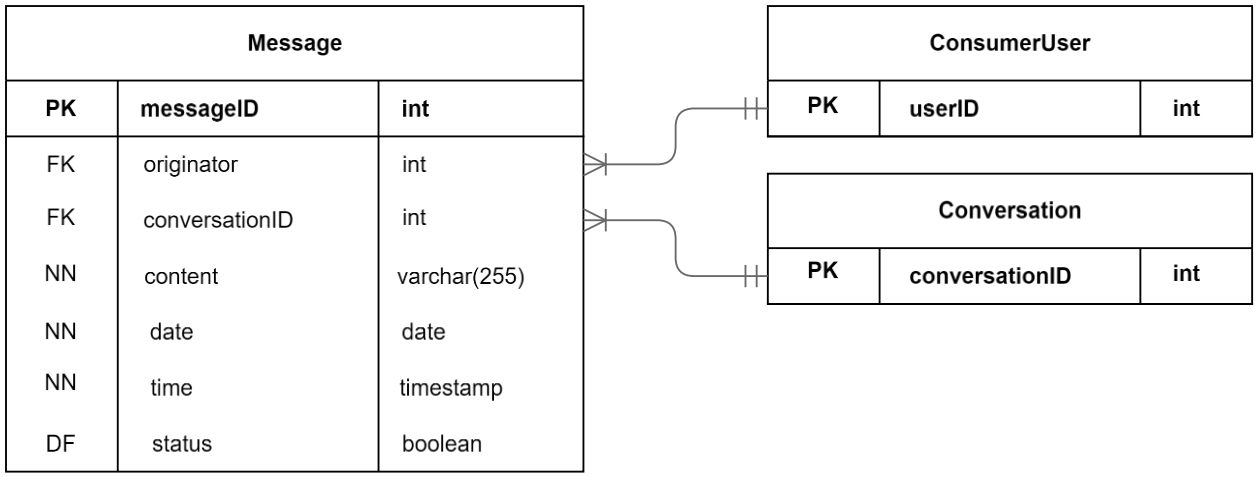
Pseudocode

```
CREATE TYPE reportType_enum AS ENUM (
    'no show', 'spam', 'fake'
);

CREATE TABLE userReport (
    reportID serial4 NOT NULL,
    reportingUser int4 NOT NULL,
    reportedUser int4 NOT NULL,
    reportType reportType_enum NOT NULL,
    eventID int4,
    CONSTRAINT reportID_pkey PRIMARY KEY (reportID),
    CONSTRAINT fk_reportingUser FOREIGN KEY (reportingUser) REFERENCES consumerUser (userID),
    CONSTRAINT fk_reportedUser FOREIGN KEY (reportedUser) REFERENCES consumerUser (userID),
    CONSTRAINT fk_eventID FOREIGN KEY (eventID) REFERENCES event (eventID)
);
```

Name	**1.1.1.7.1 UserReport SQL**
Purpose	Shows a relationship between a reported user, a report type, and an associated event (if applicable) for an admin to review.
Description	The UserReport table will store all the information regarding reports.
Requirements	3.5.7
Elements	**reportType**: options: “No Show” (LDR 7.4.1), “Spam”, or “Fake Account” (SAS.3, SAV.3).
Referenced By	1.1.1.7
Viewpoint	Pseudocode

View 1.1.1.10 Message



Name	1.1.1.10 Message
Purpose	Describe the message table fields, types, and constraints and its relationship to other tables
Description	The Message table will store all the information needed to represent a message
Requirements	3.5.10
Elements	**content**: the text of the message the user sends
	date: the date the message was sent
	time: the time the message was sent
	status: indicates if the message successfully sent, DF = false
	1.1.1.1 ConsumerUser
	1.1.1.8 Conversation
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

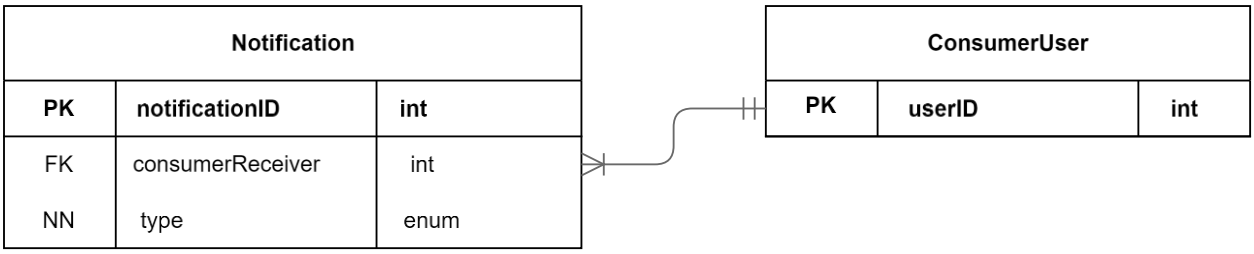
View 1.1.1.10.1 Message SQL

Pseudocode

```
CREATE TABLE message (
        messageID serial4 NOT NULL,
        originator int4 NOT NULL,
        conversationID int4 NOT NULL,
        content varchar(255) NOT NULL,
        date DATE NOT NULL,
        time TIME NOT NULL,
        status BOOLEAN DEFAULT FALSE,
        CONSTRAINT messageID_pkey PRIMARY KEY (messageID),
        CONSTRAINT fk_originator FOREIGN KEY (originator) REFERENCES consumerUser (userID)
);
```

Name	**1.1.1.10.1 Message SQL**
Purpose	Describe the message table fields, types, and constraints and its relationship to other tables
Description	The Message table will store all the information needed to represent a message
Requirements	3.5.10
Elements	**content**: the text of the message the user sends
	date: the date the message was sent
	time: the time the message was sent
	status: indicates if the message successfully sent
Referenced By	1.1.1.10
Viewpoint	Pseudocode

View 1.1.1.11 Notification



Name	1.1.1.11 Notification
Purpose	Describe the notification table fields, types, and constraints and its relationship to other tables
Description	The Notification Table will store all the information needed to represent a notification
Requirements	3.5.11
Elements	**type**: options: “Commercial” (PO.1.7.29), “Issue” (PO.1.7.41), “Consumer-Message” (URS.1.6), “Consumer-Event” (URS.1.7), “Consumer-Friend” (URS.1.8), “Consumer-Group” (URS.1.9), “Consumer-Nearby” (URS.1.10), “Admin-Message” (URS.1.11), “Admin-Event” (URS.1.12), “Admin-Commercial-Created" (URS.1.13), “Admin-Flagged” (FS.36.1), “Admin-Ticket” (FS.36.2)
	1.1.1.1 ConsumerUser
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.11.1 Notification SQL

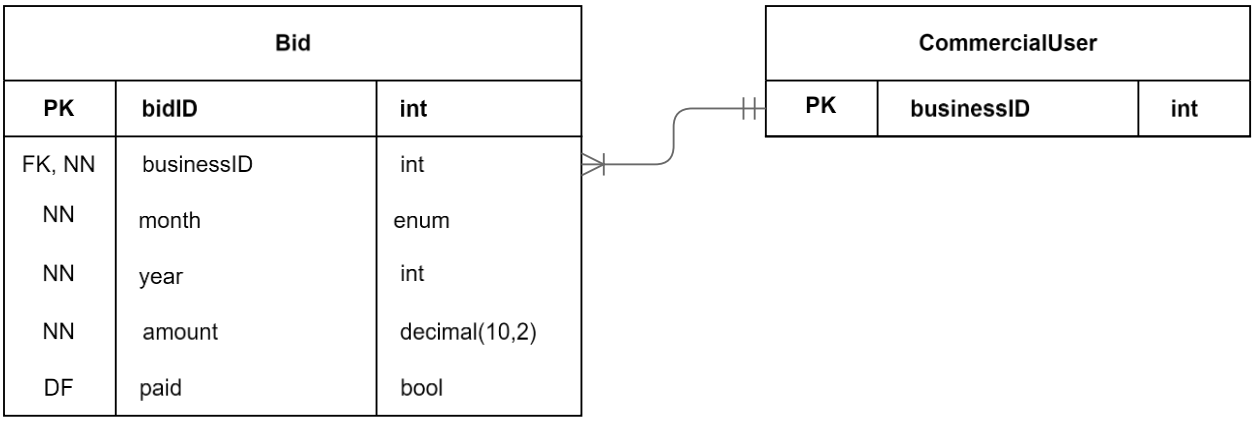
Pseudocode

```
CREATE TYPE type_enum AS ENUM (
      'Commercial', 'Issue', 'Consumer-Message', 'Consumer-Event', 'Consumer-Friend', 'Consumer-Group',
      'Consumer-Nearby', 'Admin-Message', 'Admin-Event', 'Admin-Commercial-Created', 'Admin-Flagged', 'Admin-Ticket'
);

CREATE TABLE notification (
   notificationID serial4 NOT NULL,
   consumerReceiver int4 NOT NULL,
   type type_enum NOT NULL,
   CONSTRAINT notificationID_pkey PRIMARY KEY (notificationID),
   CONSTRAINT fk_consumerReceiver FOREIGN KEY (consumerReceiver) REFERENCES consumerUser(userID)
);
```

Name	**1.1.1.11.1 Notification SQL**
Purpose	Provide the SQL for generating the Notification table
Description	The Notification SQL details all fields, types, and constraints that make up the Notification table.
Requirements	3.5.11
Elements	
Referenced By	1.1.1.11
Viewpoint	Pseudocode

View 1.1.1.12 Bid



Name	1.1.1.12 Bid
Purpose	Describes the Bid fields, types, and constraints and their relationship to other tables. For Promote Event (FV.23) and Start Auction (FV.23.4).
Description	Bid will store all the information needed to represent a given Bid for a given month (FS.23.4). Bids are held monthly (PO.3.2) and the highest bids for the month have the most advertising space and frequency.
Requirements	3.5.12
Elements	**paid**: DF = false
	1.1.1.2 CommercialUser
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.12.1 Bid SQL

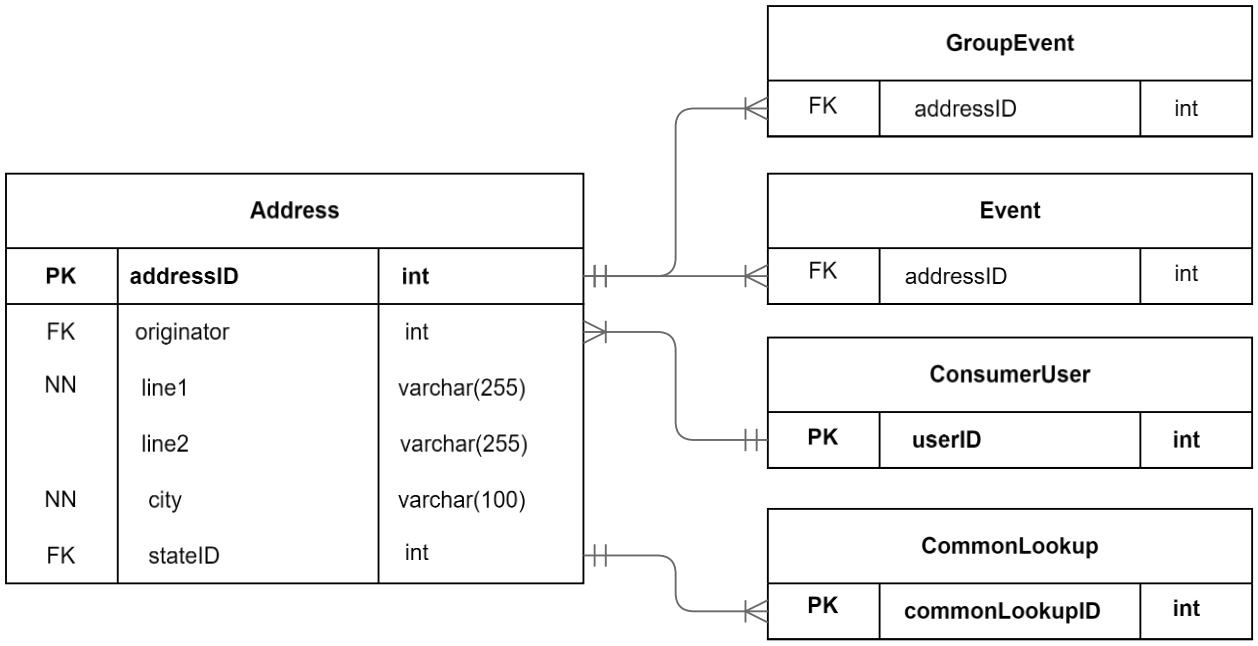
Pseudocode

```
CREATE TYPE month_enum AS ENUM (
      'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
);

CREATE TABLE bid (
   bidID serial4 NOT NULL,
   businessID int4 NOT NULL,
   month month_enum NOT NULL,
   year int4 NOT NULL,
   amount DECIMAL(10, 2) NOT NULL,
   paid BOOLEAN NOT NULL DEFAULT FALSE,
   CONSTRAINT bidID_pkey PRIMARY KEY (bidID),
   CONSTRAINT fk_businessID FOREIGN KEY (businessID) REFERENCES commercialUser (businessID)
);
```

Name	**1.1.1.12.1 Bid SQL**
Purpose	Provide the SQL for generating the Bid table
Description	The Bid SQL details all fields, types, and constraints that make up the Bid table
Requirements	3.5.13
Elements	
Referenced By	1.1.1.12
Viewpoint	Pseudocode

View 1.1.1.13 Address



Name	1.1.1.13 Address
Purpose	Describes the Address fields, types, and constraints and its relationship to other tables. Stores the addresses for both consumer (GroupEvent) and commercial (Event) events.
Description	Address will store all the information needed to represent an address for Event or GroupEvent.
Requirements	3.5.13
Elements	**originator**: Consumer Users can be either Consumer or Commercial
	1.1.1.3 Event
	1.1.1.5 GroupEvent
	MISSING: **CommonLookup**: Does not exist (LDR.13.8, LDR.14.3)
	1.1.1.1 ConsumerUser
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.13.1 Address SQL

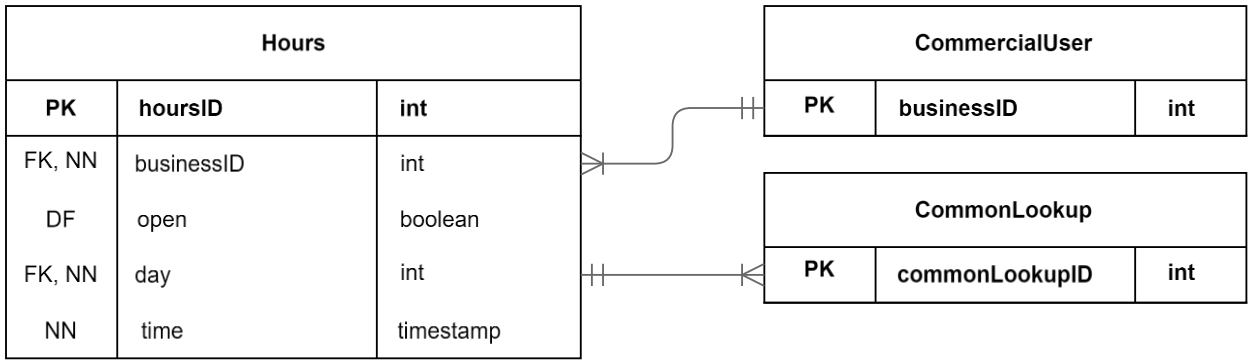
Pseudocode

```
CREATE TYPE us_state_enum AS ENUM (
    'AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DE', 'FL', 'GA', 'HI', 'ID', 'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD', 'MA', 'MI',
    'MN', 'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM', 'NY', 'NC', 'ND', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN',
    'TX', 'UT', 'VT', 'VA', 'WA', 'WV', 'WI', 'WY'
);

CREATE TABLE address (
    addressID serial4 NOT NULL,
    originator int4 NOT NULL
    street varchar(255) NOT NULL,
    city varchar(100) NOT NULL,
    state us_state_enum NOT NULL,
    zip varchar(10) NOT NULL,
    CONSTRAINT addressID_pkey PRIMARY KEY (addressID)
    CONSTRAINT fk_originator FOREIGN KEY (originator) REFERENCES consumerUser (userID)
);
```

Name	**1.1.1.13.1 Address SQL**
Purpose	Describes the Address fields, types, and constraints and its relationship to other tables. Stores the addresses for both consumer (GroupEvent) and commercial (Event) events.
Description	Address will store all the information needed to represent an address for Event or GroupEvent.
Requirements	3.5.13
Elements	**originator**: Consumer Users can be either Consumer or Commercial
Referenced By	1.1.1
Viewpoint	Pseudocode

View 1.1.1.14 Hours



Name	1.1.1.14 Hours
Purpose	Describe the Hours fields, types, and constraints and its relationship to other tables
Description	The Hours Table will store the hours of the events.
Requirements	3.5.14
Elements	**open**: List whether a slot is open to be booked for an event
	time: List the time of the event
	day: List open days that events can take place on.
	1.1.1.2 CommercialUser
Referenced By	1.1.1
Viewpoint	Entity Relationship Diagram

View 1.1.1.14.1 Hours SQL

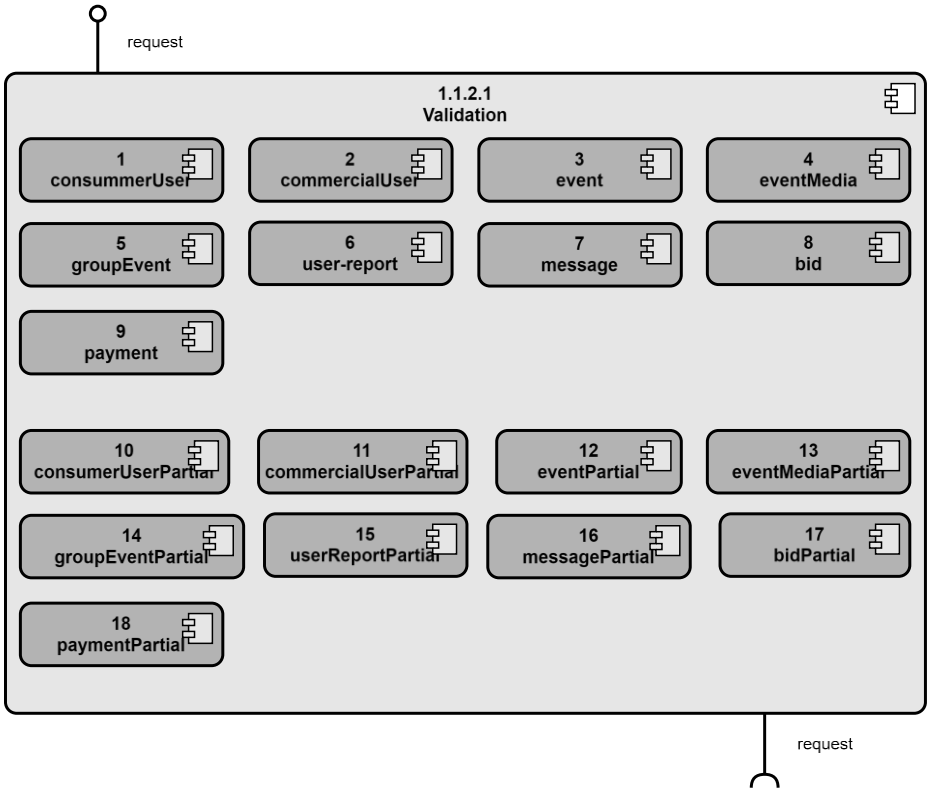
Pseudocode

```
CREATE TYPE day_enum AS ENUM (
   'M', 'T', 'W', 'R', 'F', 'SA', 'SU'
);

CREATE TABLE hours (
   hoursID serial4 NOT NULL,
   businessID int4 NOT NULL,
   open BOOLEAN DEFAULT FALSE,
   day day_enum NOT NULL,
   time TIME,
   CONSTRAINT hoursID_pkey PRIMARY KEY (hoursID),
   CONSTRAINT fk_businessID FOREIGN KEY (businessID) REFERENCES commercialUser (businessID)
);
```

Name	**1.1.1.14.1 Hours SQL**
Purpose	Describe the Hours fields, types, and constraints and its relationship to other tables
Description	The Hours SQL will create the hours table and its relationships.
Requirements	3.5.14
Elements	**open**: List whether a slot is open to be booked for an event
	time: List the time of the event
	day: List open days that events can take place on.
Referenced By	1.1.1.14
Viewpoint	Pseudocode

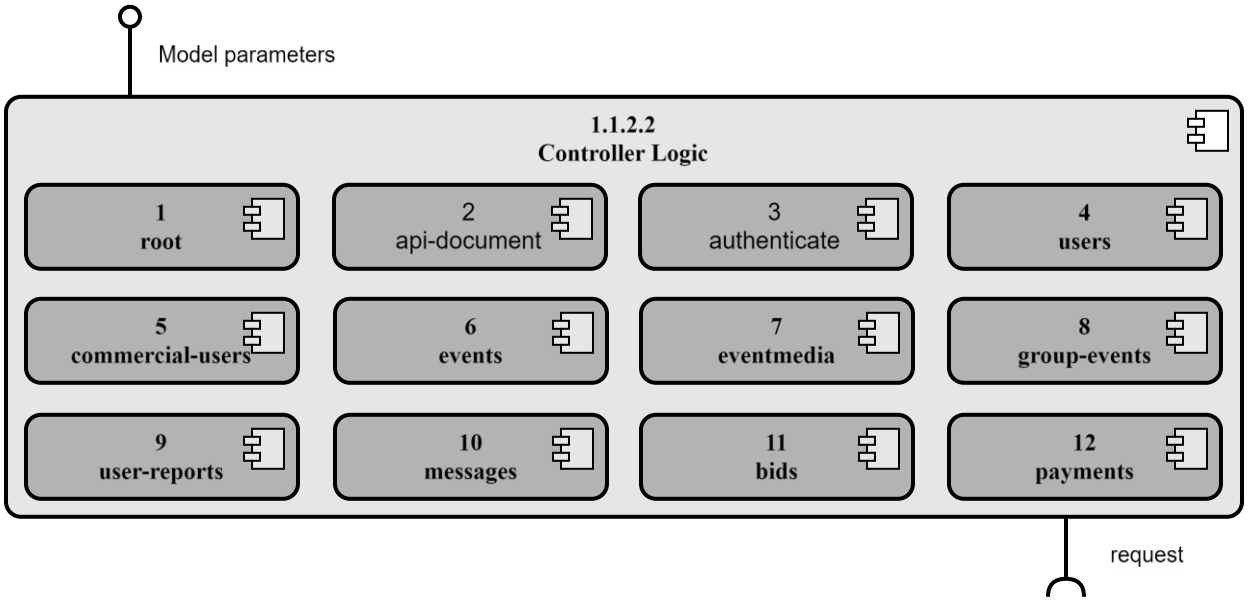
View 1.1.2.1: Validation CD



Name	**1.1.2.1 Validation**
Purpose	Validators for various data payloads the system will process.
Description	This is a collection of validator middleware functions to test request body and parameter data.
Requirements	FS.1, FS.8, FS.9, FS.10, FS.11, FS.14, FS.15, FS.16, FS.17, FS.18, FS.19, FS.20, FS.21, FS.23, FS.25, FS.26, FS.27, FS.28, FS.29, FS.30, FS.3, FS.32, FS.33, FS.34, FS.36, FS.38, FS.39, FS.40
Elements	**1.1.2.1.1 consumerUser** Model for consumer user data
	1.1.2.1.2 commercialUser Model for commercial user data
	1.1.2.1.3 event Model for event data
	1.1.2.1.4 eventMedia Model for event media data
	1.1.2.1.5 groupEvent Model for group event data
	1.1.2.1.6 userReport Model for user report data
	1.1.2.1.7 message Model for message data
	1.1.2.1.8 bid Model for bid data
	1.1.2.1.9 payment Model for payment data
	1.1.2.1.10 consumerUserPartial Model for partial user data, without required test.
	1.1.2.1.11 commercialUserPartial Model for partial commercial user data, without required test.
	1.1.2.1.12 eventPartial Model for partial event data, without required test.
	1.1.2.1.13 eventMediaPartial Model for partial event media, without required test.

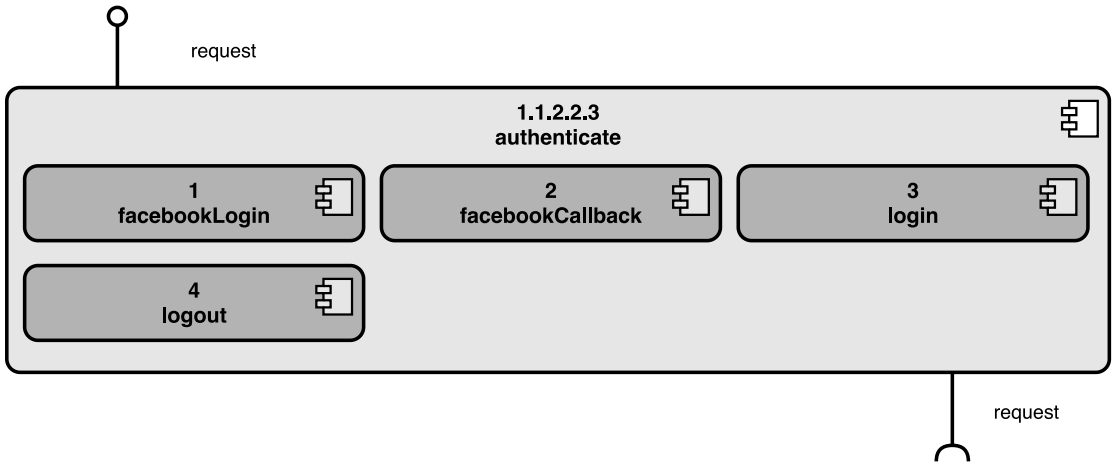
	1.1.2.1.14 groupEventPartial Model for partial group event data, without required test.
	1.1.2.1.15 userReportPartial Model for user report data, without required test.
	1.1.2.1.16 messagePartial Model for partial message data, without required test.
	1.1.2.1.17 bidPartial Model for partial bid data, without required test.
	1.1.2.1.18 paymentPartial Model for partial payment data, without required test.
Referenced by	1.1
Viewpoint	Component Diagram

View 1.1.2.2: Controller Logic Component Diagram



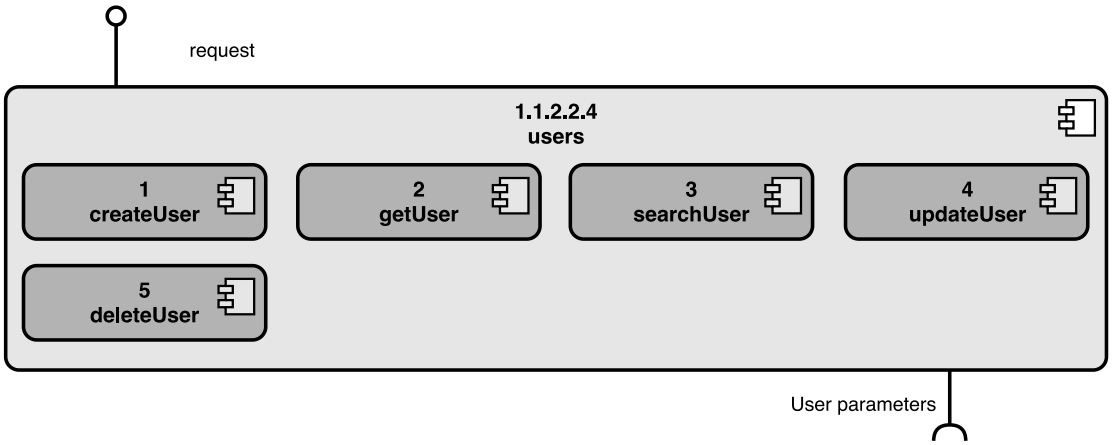
Name	1.1.2.2 Controller Logic
Purpose	The controller logic is a collection of functions to handle various API endpoint tasks
Description	The controller logic is the last step on the middleware pipeline before the database is accessed and JSON is returned to the client. Any special logic or processing is performed here.
Requirements	FS.1, FS.8, FS.9, FS.10, FS.11, FS.14, FS.15, FS.16, FS.17, FS.18, FS.19, FS.20, FS.21, FS.23, FS.25, FS.26, FS.27, FS.28, FS.29, FS.30, FS.3, FS.32, FS.33, FS.34, FS.36, FS.38, FS.39, FS.40
Elements	**1.1.2.2.1 Root.** This route serves the web client to the browser.
	1.1.2.2.2 API-document Function to serve api documentation
	1.1.2.2.3 authenticate Function to handle user authentication
	1.1.2.2.4 **users** Function to handle user CRUD
	1.1.2.2.5 **commercial-users** Function to handle commercial user CRUD
	1.1.2.2.6 **events** Function to handle event CRUD
	1.1.2.2.7 **eventmedia** Function to handle event media CRUD
	1.1.2.2.8 **group-events** Function to handle group event CRUD
	1.1.2.2.9 **user-reports** Function to handle user report CRUD
	1.1.2.2.10 **messages** Function to handle message CRUD
	1.1.2.2.11 **bids** Function to handle bid CRUD
	1.1.2.2.12 **payments** Function to handle payment CRUD
Referenced by	1.1
Viewpoint	Component Diagram

View 1.1.2.2.3 Authentication Controller



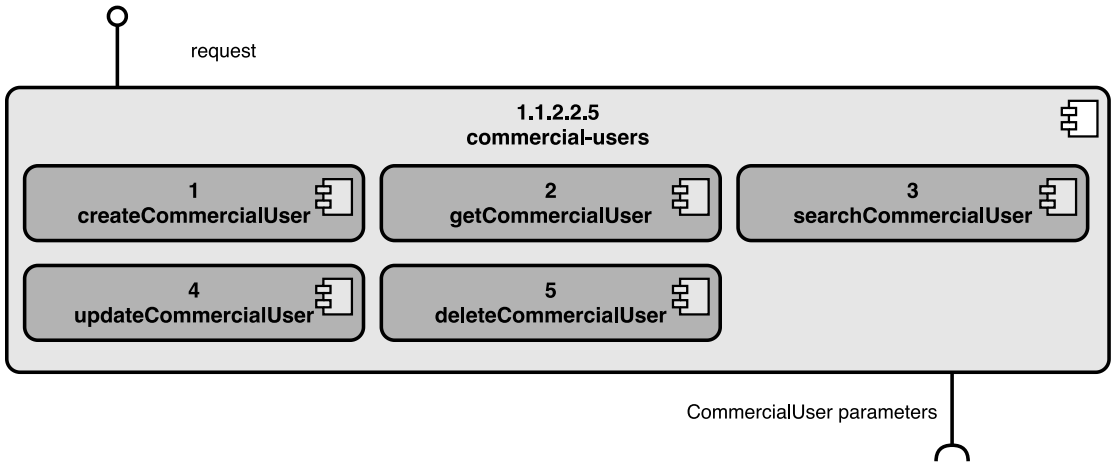
Name	1.1.2.2.3 Authentication Controller
Purpose	Endpoint controllers to handle authentication concerns
Description	This controller will handle initiating the login process and handling Facebook oauth authentication callbacks.
Requirements	FS.6, FS.14, FS.18, FS.25, FS.30, FS.38
Elements	**1.1.2.2.3.1 facebookLogin** This returns the needed credentials to the client
	1.1.2.2.3.2 facebookCallback After authorizing through facebook, this handles the callback from facebook.
	1.1.2.2.3.3 login This initiates the traditional username/password login
	1.1.2.2.3.4 logout This logs the current user out
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.4 Users Controller



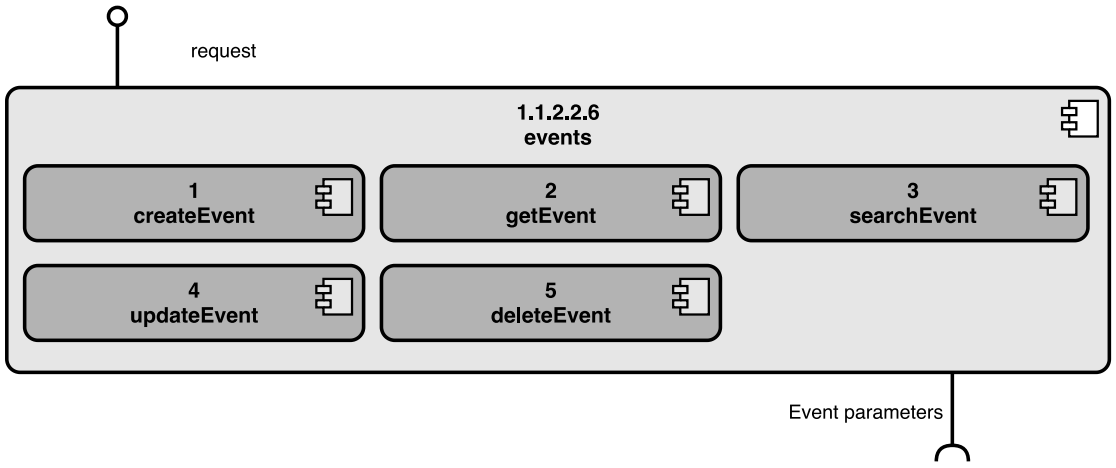
Name	1.1.2.2.4 Users Controller
Purpose	Endpoint controller to handle users
Description	This endpoint handles basic crud and search operations for users.
Requirements	SAS.2, FV.32, URV.2, URV.3, LDR.1
Elements	**1.1.2.2.2.1 createUser** Creates a user record in the database
	1.1.2.2.2.2 getUser Gets a user by id from the database
	1.1.2.2.2.3 searchUser Gets an array of users based on a search criteria
	1.1.2.2.2.4 updateUser Updates user data in the database by id
	1.1.2.2.2.5 deleteUser Deletes user data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	User paramters Data necessary to create a new User model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.5 Commercial Users Controller



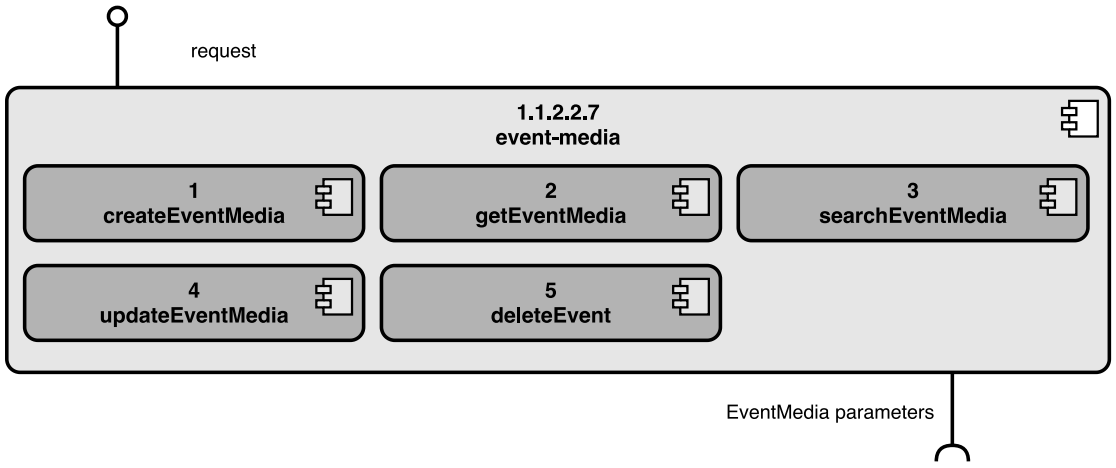
Name	1.1.2.2.5 Commercial Users Controller
Purpose	Endpoint controller to handle commercial users
Description	This endpoint handles basic crud and search operations for commercial user accounts.
Requirements	URS.3, PO.1.7.18, PO.2.2, PO.3.2, EIS.1.2
Elements	**1.1.2.2.5.1 createCommercialUser** Creates a commercial user record in the database
	1.1.2.2.5.2 getCommercialUser Gets a commercial user by id from the database
	1.1.2.2.5.3 searchCommercialUser Gets an array of commercial users based on a search criteria
	1.1.2.2.5.4 updateCommercialUser Updates commercial user data in the database by id
	1.1.2.2.5.5 deleteCommercialUser Deletes commercial user data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	CommercialUser parameters Data to create a new CommercialUser model
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.6 Events Controller



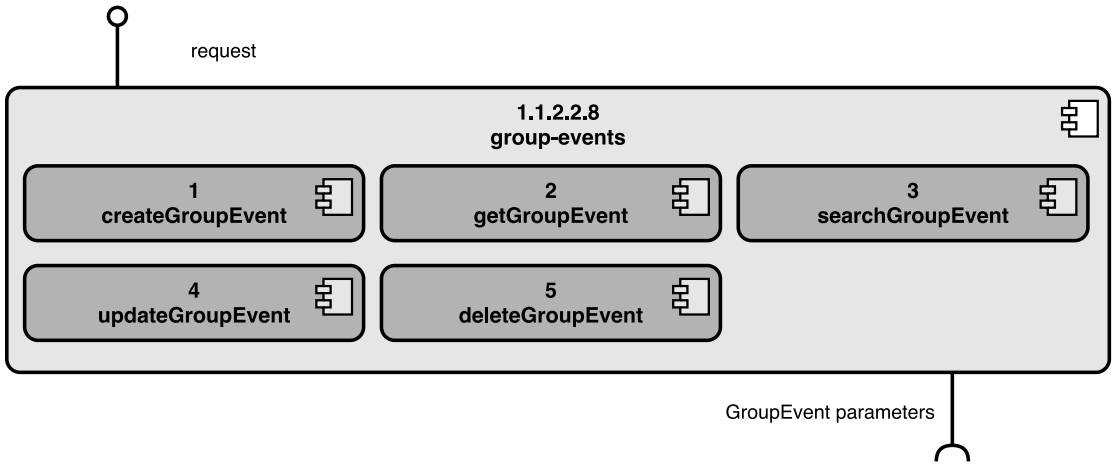
Name	1.1.2.2.6 Event Controller
Purpose	Endpoint controller to handle events
Description	The endpoint handles basic crud and search operations for events
Requirements	LDR.3, FV.7, FV.7.2, FV.11, FV.21, FV.33, URV.1.10, URV.4.5
Elements	**1.1.2.2.6.1 createEvent** Creates a event record in the database
	1.1.2.2.6.2 getEvent Gets a event by id from the database
	1.1.2.2.6.3 searchEvent Gets an array of events based on a search criteria
	1.1.2.2.6.4 updateEvent Updates event data in the database by id
	1.1.2.2.6.5 deleteEvent Deletes event data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	Event parameters Data necessary to create a new Event model object
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.7 Event Media Controller



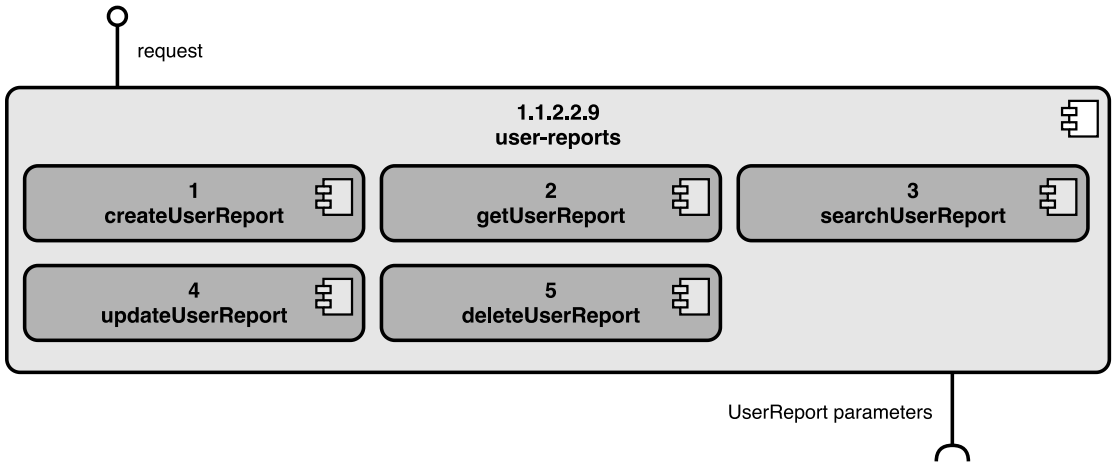
Name	1.1.2.2.7 Event Media Controller
Purpose	Endpoint controller to handle event media
Description	This endpoint handles basic crud and search operations for bids.
Requirements	LDR.4, LDV.4, LDV.4.2, LDV.4.3, SAV.3.3
Elements	**1.1.2.2.7.1 createEventMedia** Creates a bid record in the database
	1.1.2.2.7.2 getEventMedia Gets a bid by id from the database
	1.1.2.2.7.3 searchEventMedia Gets an array of bids based on a search criteria
	1.1.2.2.7.4 updateEventMedia Updates bid data in the database by id
	1.1.2.2.7.5 deleteEventMedia Deletes bid data from the databse by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	EventMedia parameters Data necessary to create a new EventMedia model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.8 Group Event Controller



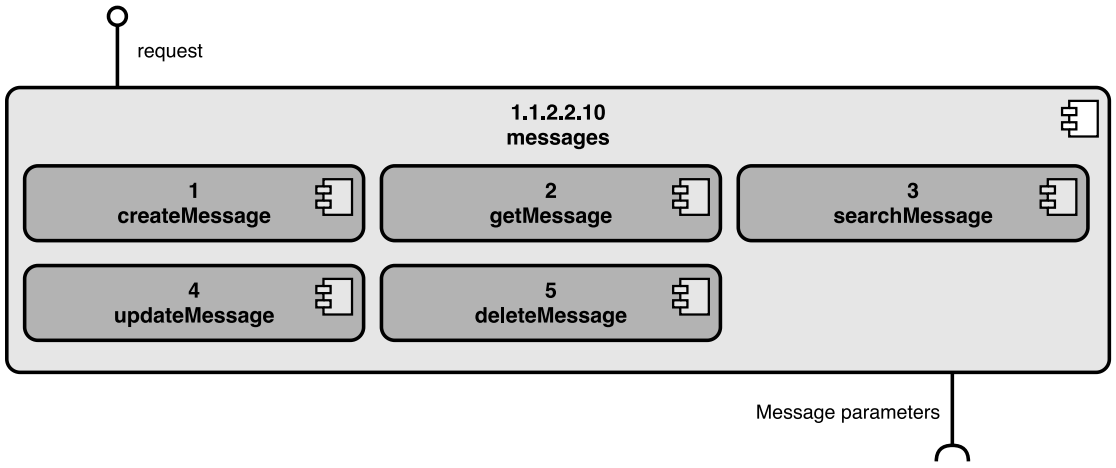
Name	1.1.2.2.8 Group Event Controller
Purpose	Endpoint controller to handle group events
Description	This endpoint handles basic crud and search operations for group events.
Requirements	URV.4, URV.4.6, URV.4.7, URV.4.9, URV.4.11, PRV.1.1, LDV.5, LDV.6, PO.1.7.25, PO.2.1.3, PO.3.4.3,
Elements	**1.1.2.2.8.1 createGroupEvent** Creates a group event record in the database
	1.1.2.2.8.2 getGroupEvent Gets a group event by id from the database
	1.1.2.2.8.3 searchGroupEvent Gets an array of group events based on a search criteria
	1.1.2.2.8.4 updateGroupEvent Updates group event data in the database by id
	1.1.2.2.8.5 deleteGroupEvent Deletes group event data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	GroupEvent parameters Data necessary to create a new GroupEvent model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.9 User Reports Controller



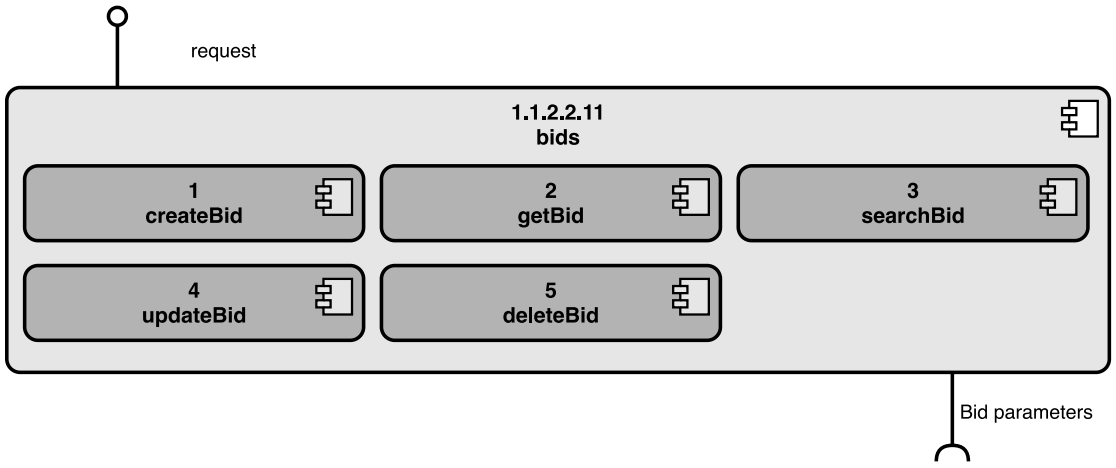
Name	1.1.2.2.9 User Reports Controller
Purpose	Endpoint controller to handle user reports
Description	This endpoint handles basic crud and search operations for user reports.
Requirements	LDR.7
Elements	**1.1.2.2.9.1 createUserReport** Creates a user report record in the database
	1.1.2.2.9.2 getUserReport Gets a user report by id from the database
	1.1.2.2.9.3 searchUserReport Gets an array of user report based on a search criteria
	1.1.2.2.9.4 updateUserReport Updates user report data in the database by id
	1.1.2.2.9.5 deleteUserReport Deletes user report data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	UserReport parameters The data necessary to create a new UserReport model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.10 Messages Controller



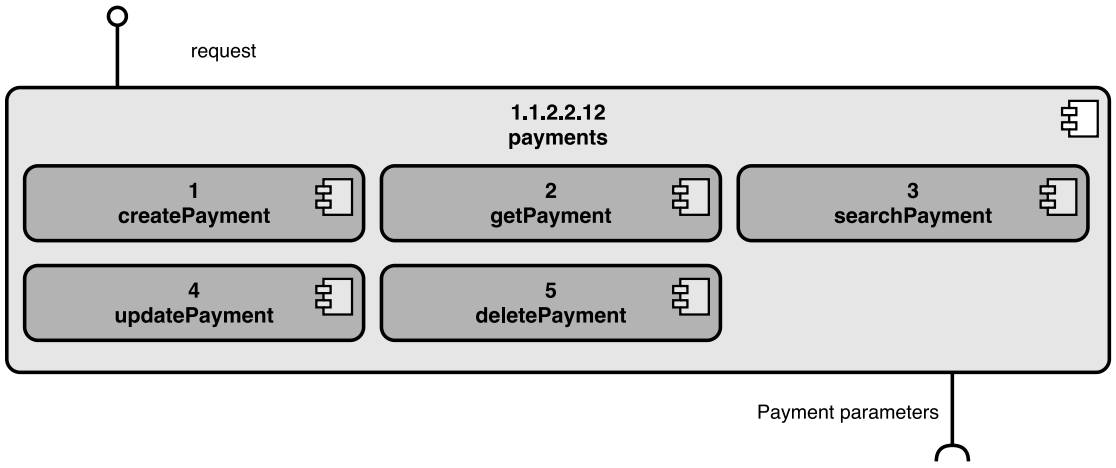
Name	1.1.2.2.10 Messages Controller
Purpose	Endpoint controller to handle messages
Description	This endpoint handles basic crud and search operations for messages.
Requirements	LDR.10, DCS.1.3, FV.10, FV.34, URV.1.6, URV.1.11, LDV.10, PO.1.7.7, PO.1.7.36, PO.3.4.2, FS.10.4, FS.34, LDR.10,
Elements	**1.1.2.2.10.1 createMessage** Creates a message record in the database
	1.1.2.2.10.2 getMessage Gets a message by id from the database
	1.1.2.2.10.3 searchMessage Gets an array of messages based on a search criteria
	1.1.2.2.10.4 updateMessage Updates message data in the database by id
	1.1.2.2.10.5 deleteMessage Deletes message data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	Message parameters Data necessary to create a new Messages model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.11 Bids Controller



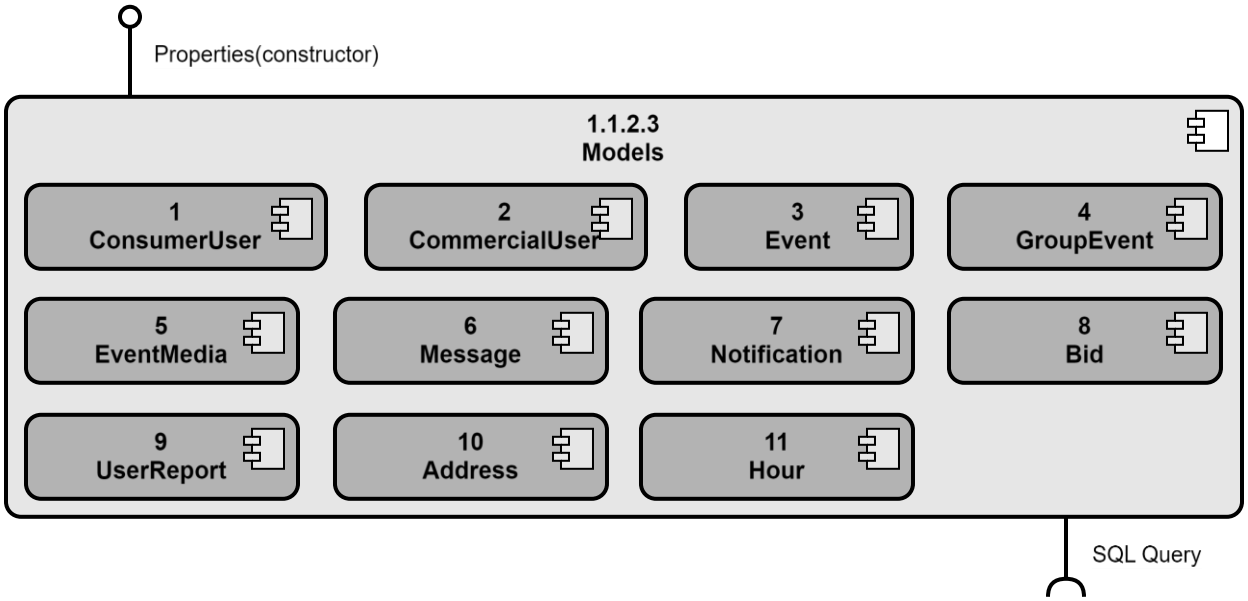
Name	1.1.2.2.11 Bids Controller
Purpose	Endpoint controller to handle auction bids
Description	This endpoint handles basic crud and search operations for bids.
Requirements	FS.23.4 [PO.2.2.17], LDR.12, FV.23
Elements	**1.1.2.2.11.1 createBid** Creates a bid record in the database
	1.1.2.2.11.2 getBid Gets a bid by id from the database
	1.1.2.2.11.3 searchBid Gets an array of bids based on a search criteria
	1.1.2.2.11.4 updateBid Updates bid data in the database by id
	1.1.2.2.11.5 deleteBid Deletes bid data from the databse by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	Bid Parameters Data about the bid to create a bid model
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.2.12 Payment Controller



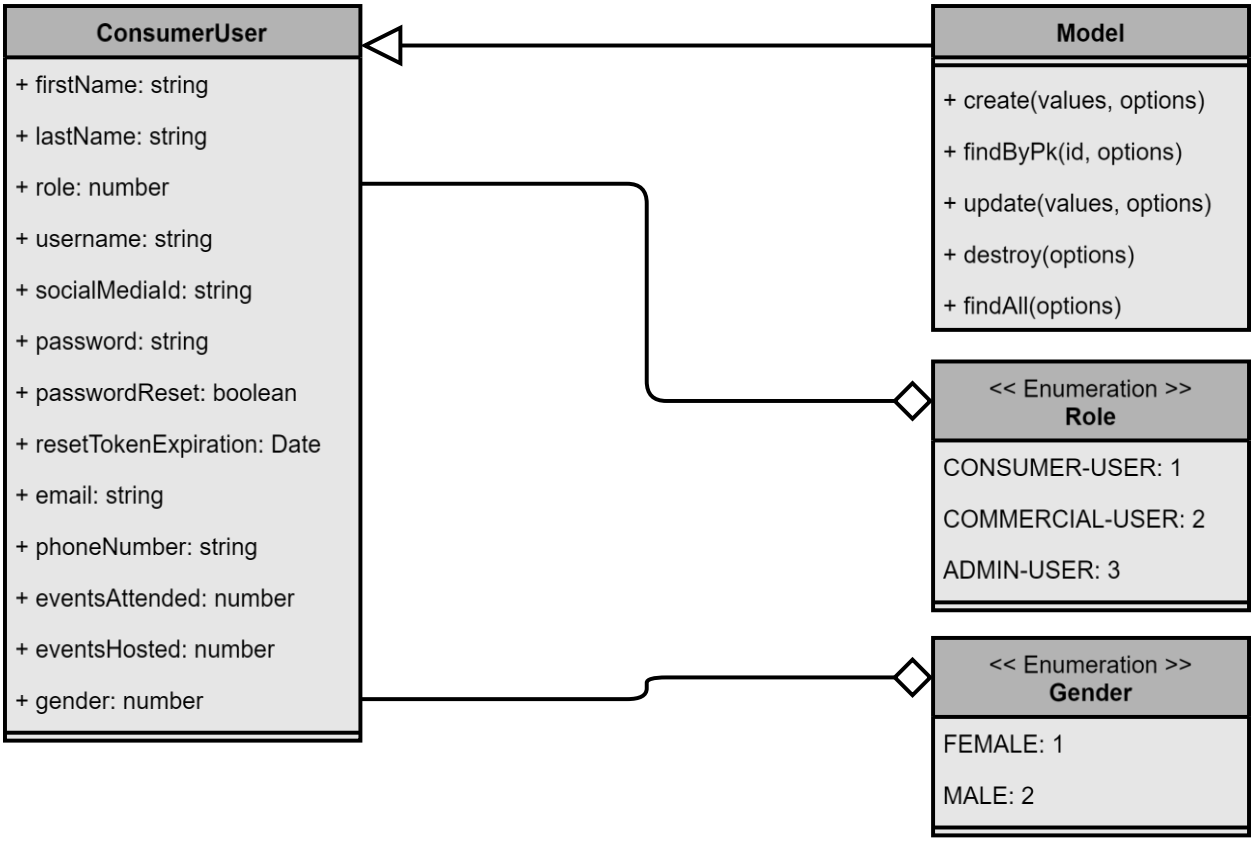
Name	1.1.2.2.12 Payment Controller
Purpose	Endpoint controller to handle payments
Description	This endpoint handles basic crud and search operations payments.
Requirements	F.15.5, FS.16.2, DCS.2, FV.15.5, FV.16.2, DCV.2, F.15.5, FS.16.2,
Elements	**1.1.2.2.12.1 createPayment** Creates a payment record in the database
	1.1.2.2.12.2 getPayment Gets a payment by id from the database
	1.1.2.2.12.3 searchPayment Gets an array of payments based on a search criteria
	1.1.2.2.12.4 updatePayment Updates payment data in the database by id
	1.1.2.2.12.5 deletePayment Deletes payment data from the database by id
	Request The standard express request object. Of special concern in the JSON body payload, potentially containing data to be processed and stored.
	Payment parameters The data necessary to create a new Payment model object.
Referenced by	1.1.2.2
Viewpoint	Component Diagram

View 1.1.2.3: Models Component Diagram



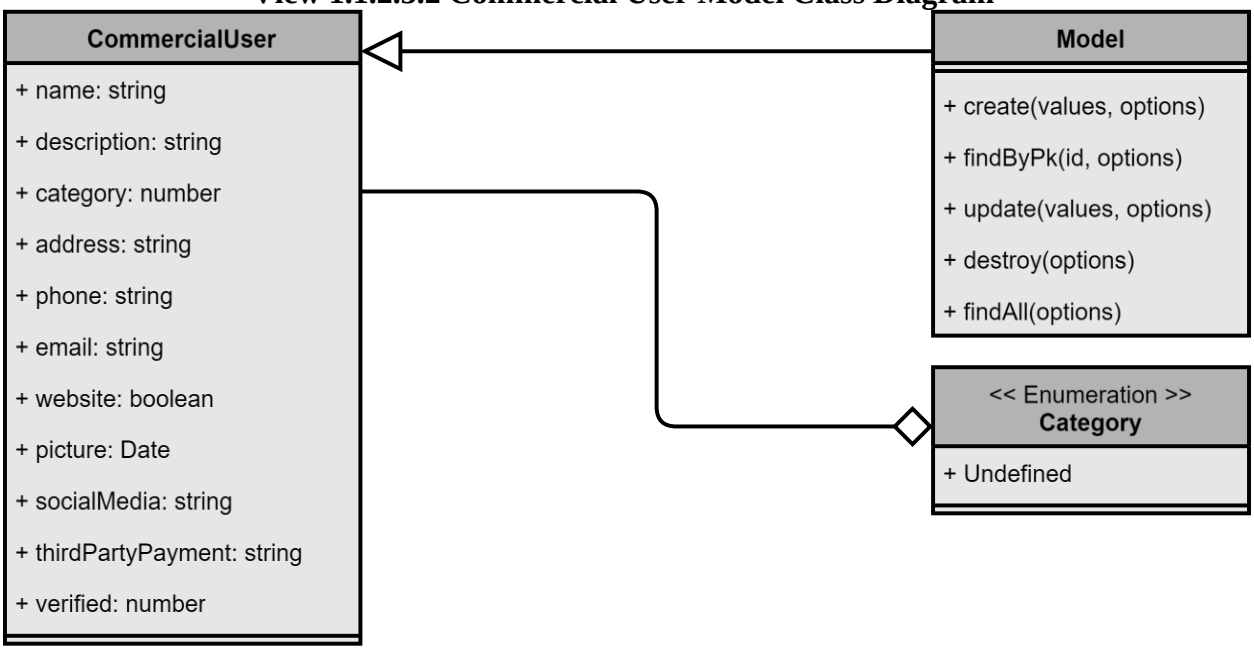
Name	**1.1.2.3 Models**
Purpose	Presents the various models the server will use to communicate with the database
Description	Each component is a Sequelize model object that interfaces with the PostgreSQL database. It handles generating SQL queries and abstracts inner joins and many to many join tables.
Requirements	FS.1, FS.8, FS.9, FS.10, FS.11, FS.14, FS.15, FS.16, FS.17, FS.18, FS.19, FS.20, FS.21, FS.23, FS.25, FS.26, FS.27, FS.28, FS.29, FS.30, FS.3, FS.32, FS.33, FS.34, FS.36, FS.38, FS.39, FS.40
Elements	1.1.2.3.1 **ConsumerUser** model
	1.1.2.3.2 **CommercialUser** model
	1.1.2.3.3 **Event** model
	1.1.2.3.4 **GroupEvent** model
	1.1.2.3.5 **EventMedia** model
	1.1.2.3.6 **Message** model
	1.1.2.3.7 **Notification** model
	1.1.2.3.8 Bid model
	1.1.2.3.9 UserReport model
	1.1.2.3.10 **Address** model
	1.1.2.3.11 **Hour** model
Referenced by	1.1
Viewpoint	Component Diagram

View 1.1.2.3.1 Consumer User Model Class Diagram



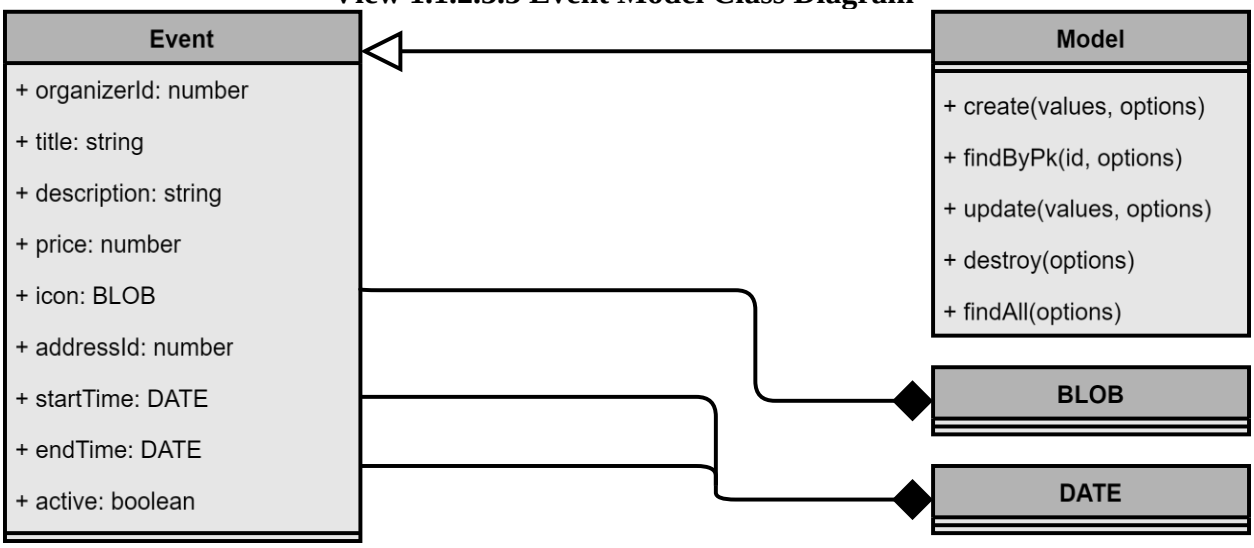
Name	1.1.2.3.1 Consumer User Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the ConsumerUser PostgreSQL table. Stores data related to consumer users using the Uvents system.
Requirements	SAS.2.5, FV.1.5, URV.2, URV.2.8
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Role JavaScript object that acts as an enumeration of user roles.
	Gender JavaScript object that acts as an enumeration of genders.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.2 Commercial User Model Class Diagram



Name	1.1.2.3.2 Commercial User Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the CommercialUser PostgreSQL table. Stores data related to commercial users using the Uvents system.
Requirements	PO.3.2, PO.3.5, FS.15
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Category (*This enumeration is currently undefined in the SRS*)
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.3 Event Model Class Diagram

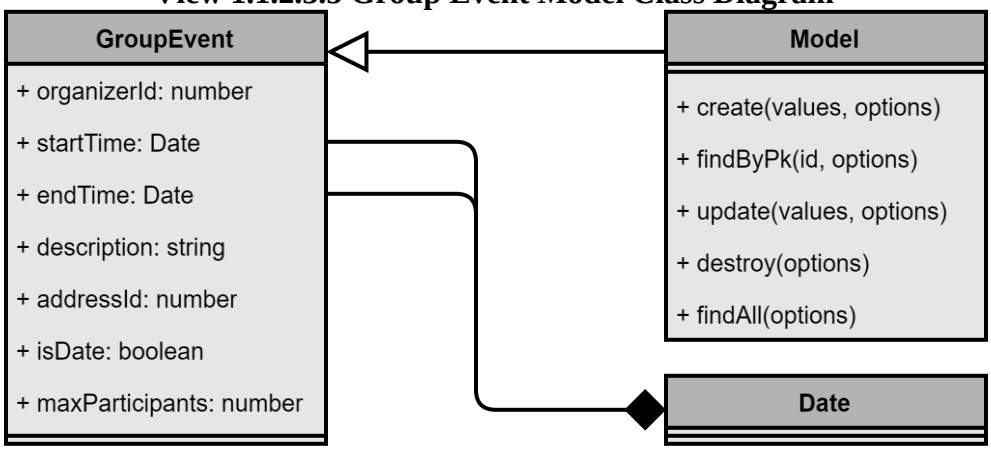


Name	1.1.2.3.3 Event Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Event PostgreSQL table. Stores data related to events created by commercial users in the Uvents system.
Requirements	PO.3.4.3, PO.3.5.3, EIS.1.4, FS.2, LDR.3
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	BLOB Sequelize.BLOB is a data type defined as a data type in the Sequelize module.
	Date The standard date/time storage class/data type in JavaScript.
	1.1.1.2 **organizerId** Index/foreign key into the CommercialUser table.
	1.1.1.13 **addressId** Index/foreign key into the Address table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.4 Event Media Model Class Diagram

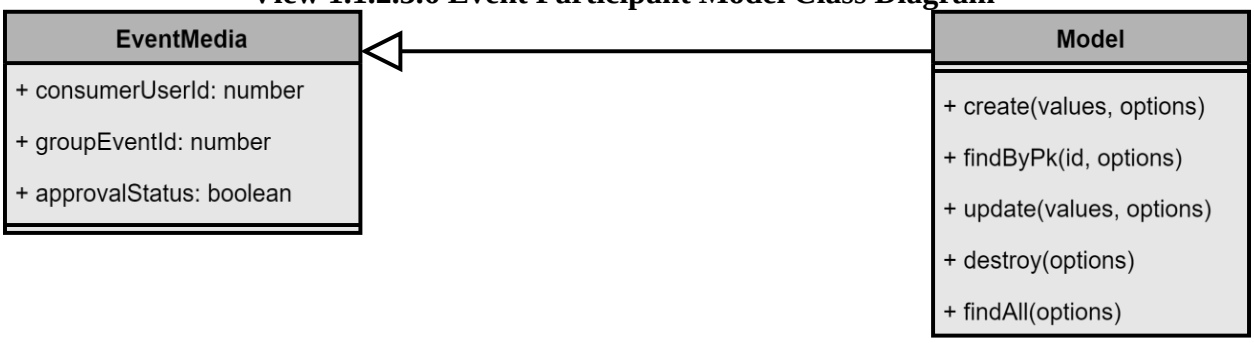
Name	1.1.2.3.4 Event Media Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the EventMedia PostgreSQL table. Stores data related to media connected to events in the Uvents system.
Requirements	LDR.4, LDV.4
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	1.1.1.3 eventId Index/foreign key into the Event database table
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.5 Group Event Model Class Diagram



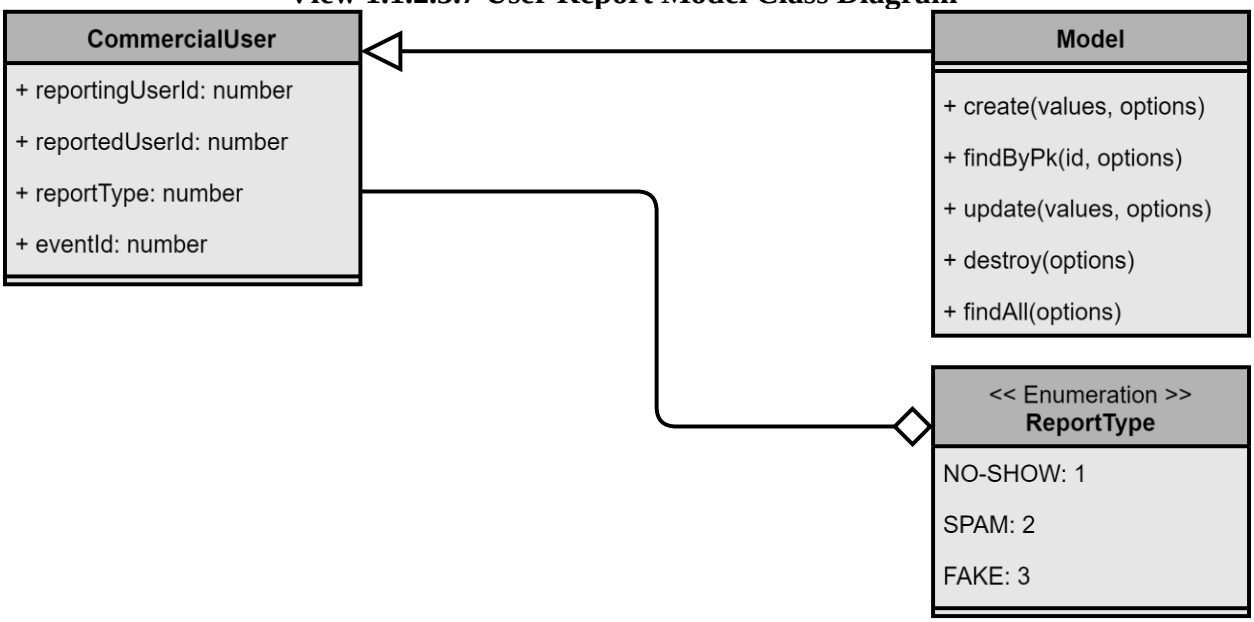
Name	1.1.2.3.5 Group Event Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Group Event PostgreSQL table. Stores data related to group events using the Uvents system.
Requirements	LDR.5, FV.3, FV.8, URV.4.6, URV.4.7, URV.4.9, LDV.5
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Date The standard date/time storage class/data type in JavaScript.
	1.1.1.2 organizerId Index/foreign key into the CommercialUser table.
	1.1.1.13 addressId Index/foreign key into the Address table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.6 Event Participant Model Class Diagram



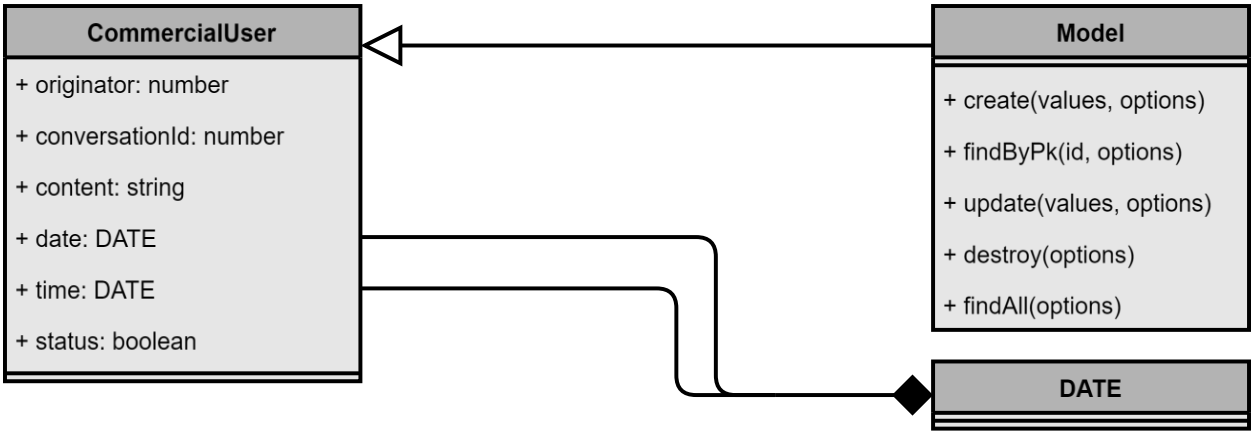
Name	1.1.2.3.6 Event Participant Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Event PostgreSQL table. Stores data related to events created by commercial users in the Uvents system.
Requirements	LDR.6, LDV.6
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	1.1.1.1 consumerUserId Index/foreign key into the ConsumerUser table.
	1.1.1.5 groupEventId Index/foreign key into the GroupEvent table
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.7 User Report Model Class Diagram



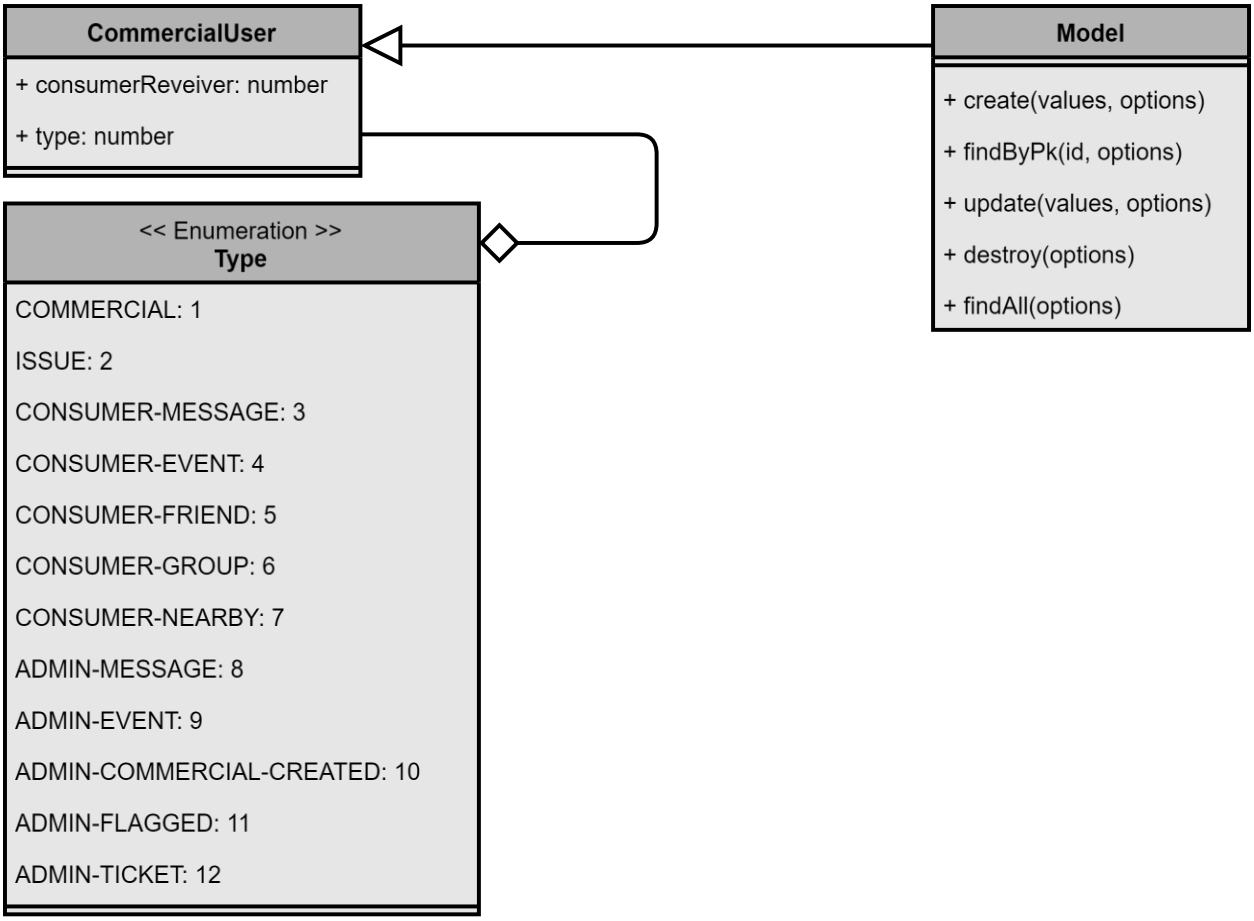
Name	1.1.2.3.7 User Report Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Event PostgreSQL table. Stores data related to events created by commercial users in the Uvents system.
Requirements	LDR.7, LDV.7
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	ReportType Enumeration of report types.
	1.1.1.1 reportingUserId Index/foreign key into the ConsumerUser table.
	1.1.1.1 reportedUserId Index/foreign key into the ConsumerUser table.
	1.1.1.3 eventId Index/foreign key into the Event table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.10 Message Model Class Diagram



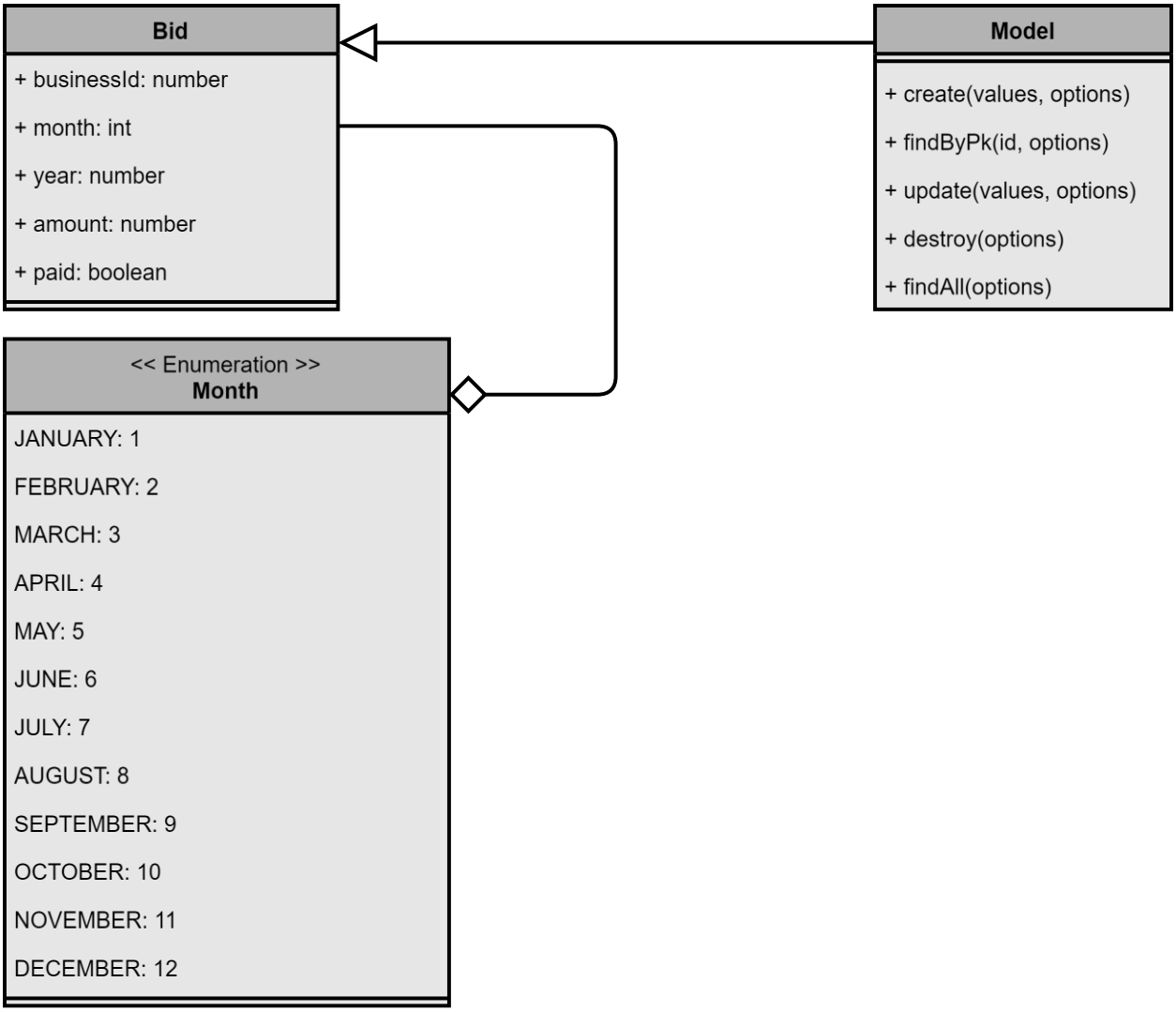
Name	1.1.2.3.10 Message Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Message PostgreSQL table. Stores data related to messages created by consumer users in the Uvents system.
Requirements	LDR.10, LDV.10, PO.1.2.5, PO.1.7.7, PO.1.7.23, PO.1.7.36, FS.10.3
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Date The standard date/time storage class/data type in JavaScript.
	1.1.1.1 originator Index/foreign key into the ConsumerUser table.
	X conversationId Index/foreign key into the Conversation table (Not defined).
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.11 Notification Model Class Diagram



Name	1.1.2.3.11 Notification Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Notification PostgreSQL table. Stores data related to notifications from the Uvents system.
Requirements	LDR.11, LDV.11, FV.12, FV.36, URV.1.6, URV.1.7-13, PO.1.2.10, PO.1.7.15, PO.1.7.29, PO.1.7.41, PO.2.1.14, PO.2.3.11, FS.12, FS.36
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Type Enumeration of the various notification types.
	1.1.1.1 consumerReceiver Index/foreign key into the ConsumerUser table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.12 Bid Model Class Diagram

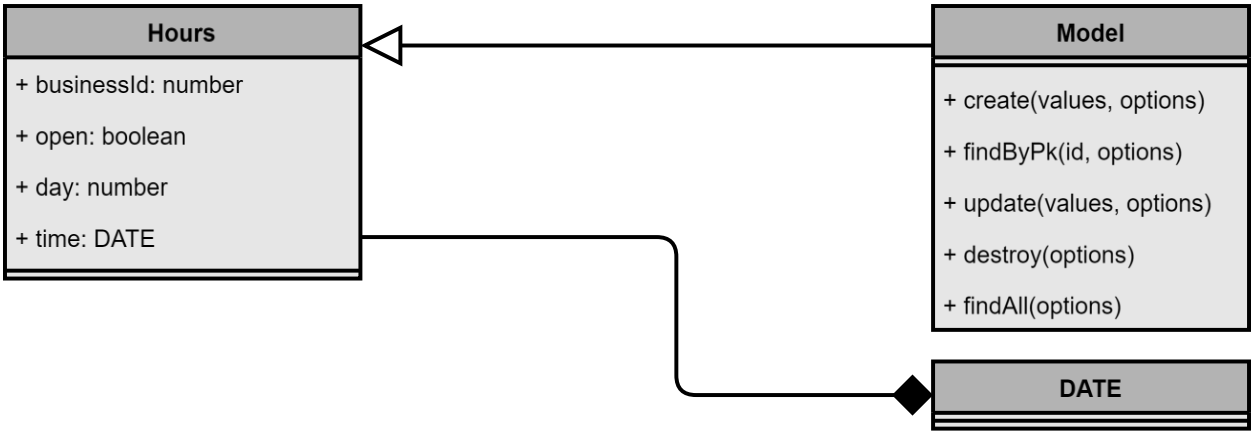


Name	1.1.2.3.12 Bid Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Bid PostgreSQL table. Stores data related to bids created by commercial users in the Uvents system.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Month Enumeration of months.
	1.1.1.2 businessId Index/foreign key into the CommercialUser table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.13 Address Model Class Diagram

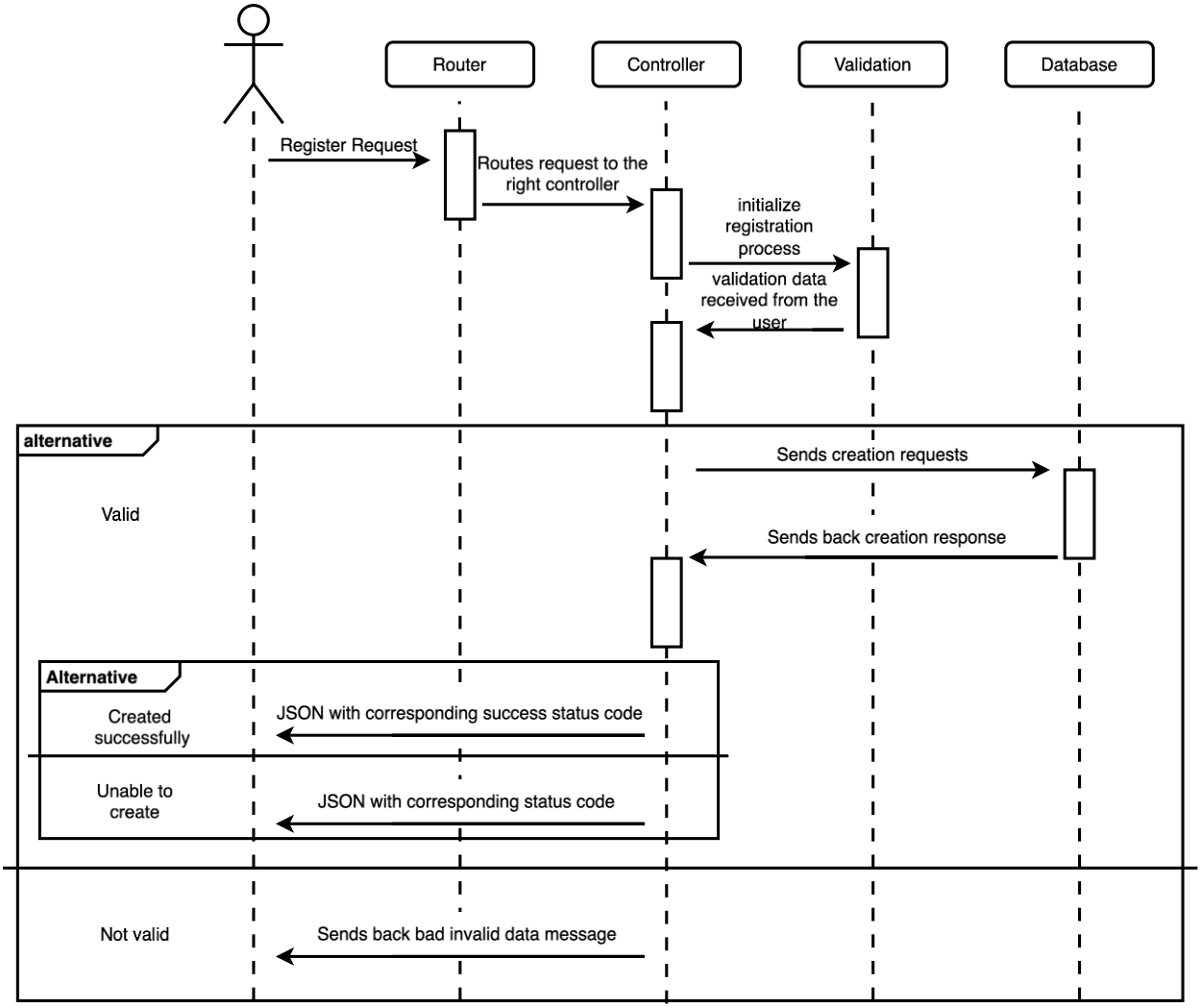
Name	1.1.2.3.13 Address Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Address PostgreSQL table. Stores data related to addresses created by commercial users in the Uvents system.
Requirements	LDR.3.7, LDR.13, LDV.13
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	1.1.1.2 originator Index/foreign key into the CommercialUser table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

View 1.1.2.3.14 Hours Model Class Diagram



Name	1.1.2.3.14 Hours Model Class Diagram
Purpose	Show the relationship between classes and objects related to sequelize database models.
Description	A model representing the Hours PostgreSQL table. Stores data related to hours created by commercial users in the Uvents system.
Requirements	LDR.14, LDV.14
Elements	**Model** A class that defines the data structure and relations of a database table, part of the Sequelize ORM module.
	Date The standard date/time storage class/data type in JavaScript.
	1.1.1.2 businessId Index/foreign key into the CommercialUser table.
Referenced by	1.1.2.3
Viewpoint	Class Diagram

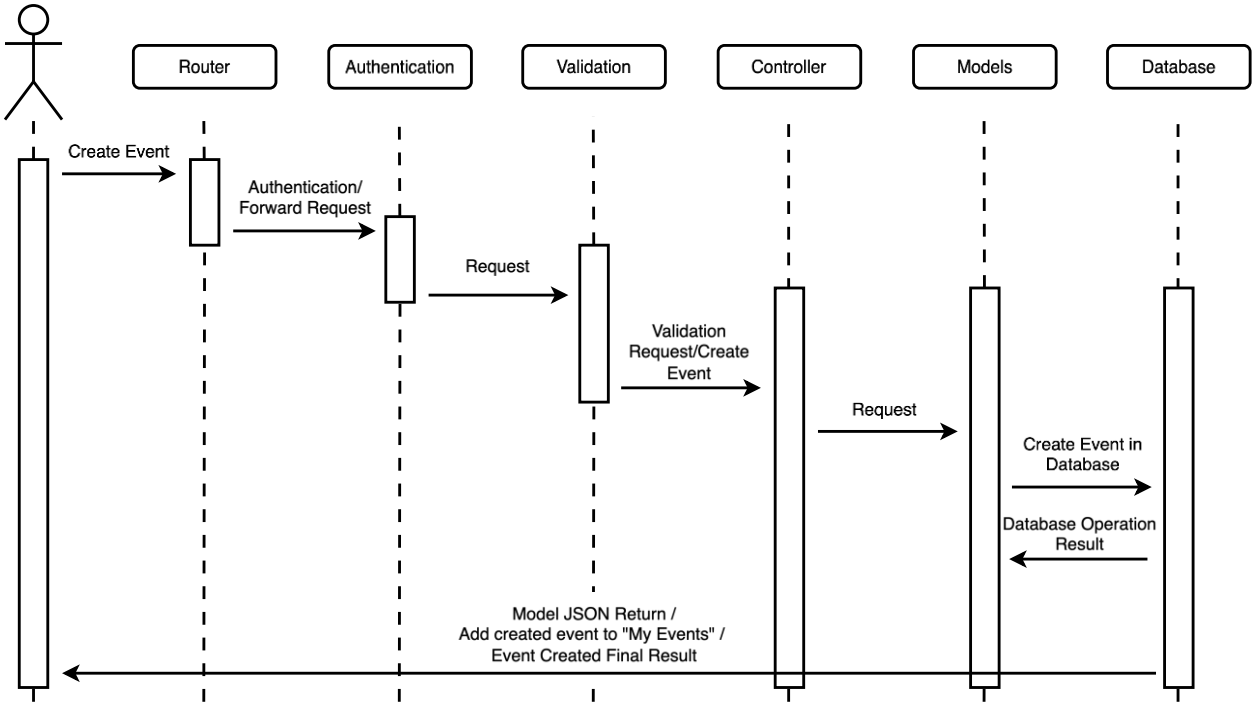
View 1.1.2.4: Register Sequence Diagram



Name	1.1.2.4: Register Sequence Diagram
Purpose	Provides the reader with an overview of the register sequence
Description	Shows the steps of a register request
Requirements	PO 1.1.2
Elements	1.1.2.1 **Register** request sent to **Router** 1.1.2.2 **Router** then routes request to the correct C**ontroller** 1.1.2.3 **Controller** initializes the registration process 1.1.2.4 Data sent by user is then validated **Server Response path** **1.1.2.5 Controller** sends creation request to the **Database** **1.1.2.6 Database** send creation response back to the **Controller** **1.1.2.7 JSON** sent to user with the corresponding success response **1.1.2.8 JSON** sent to user with the corresponding status code

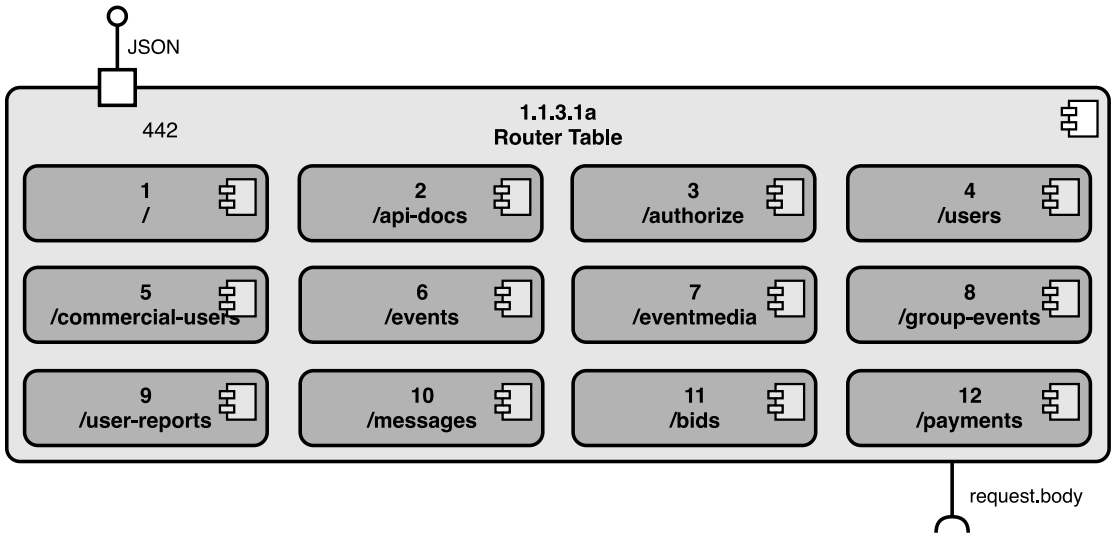
	Data Validation Failure **1.1.2.9 Controller** sends an invalid data message to the user.
Referenced by	1.1.2
Viewpoint	Sequence Diagram

View 1.1.3.1: Event Creation Sequence



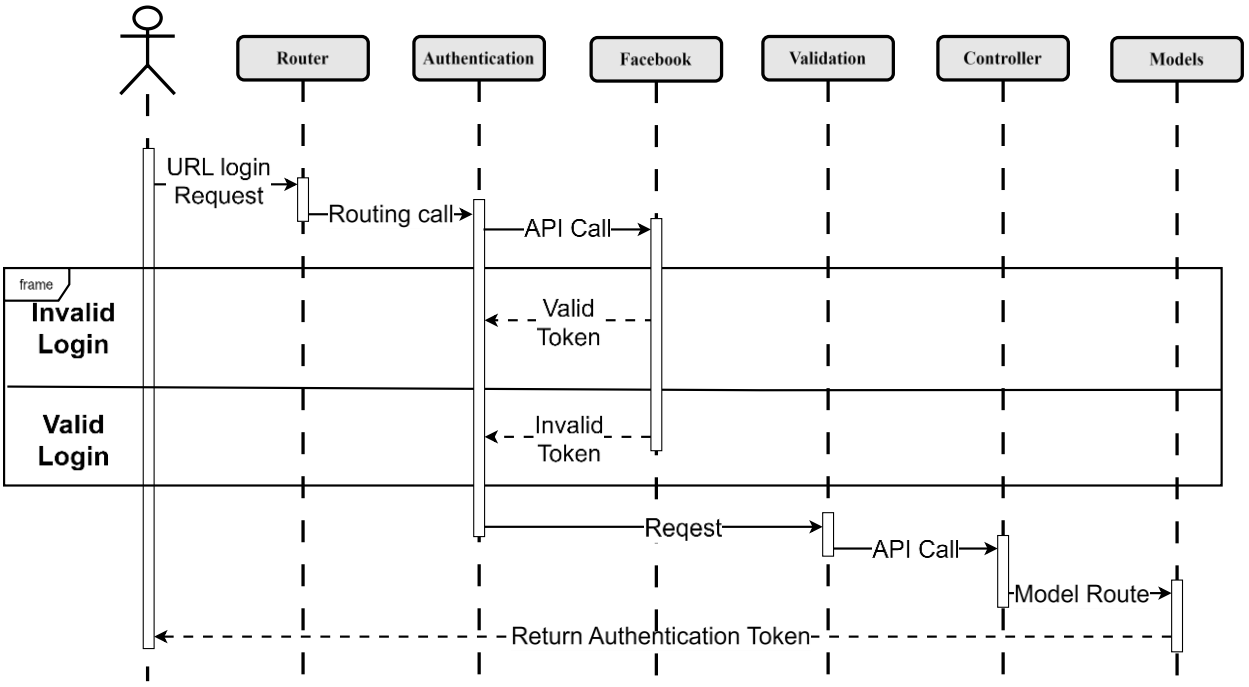
Name	**1.1.3.1 Create Event Sequence Diagram**
Purpose	Overview of the event creation by user
Description	Diagram demonstrates the sequence of step to create an event
Requirements	PO.1.1.3, PO.1.7.8, FS.8.5
Elements	1.1.3.1 An event is created by the user and is sent to the **Router** 1.1.3.2 **Authentication** and the Forward Request **1.1.3.3** The **Request** is sent from **Authentication to Validation** **1.1.3.4** The **Validated Request** is then sent along with the **Create Event** to the C**ontroller** **1.1.3.5** The **Request** is then sent from the **Controller** to the **Models** **1.1.3.6** The **Event** is then created in the **Database** and the **Database** returns the operation result **1.1.3.7** Finally the **Model JSON** return / Add created event to “My Events” and the Result are sent from the **Database** to the **Model**, then to the **Controller** and finally to the End User.
Referenced by	1.1.3
Viewpoint	Sequence Diagram

View 1.1.3.1a: Router Table Component Diagram



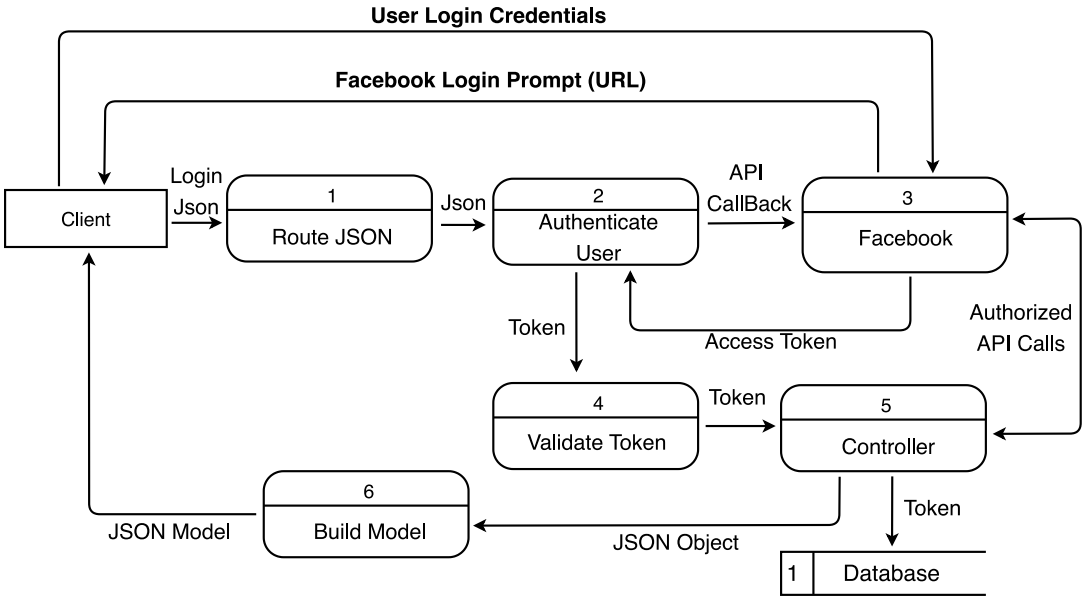
Name	1.1.3.1a Router Table
Purpose	To show the available router components and endpoints of the API.
Description	The router receives requests with optional id and JSON payload. These values are forwarded through the middleware chain to be processed.
Requirements	FS.1, FS.8, FS.9, FS.10, FS.11, FS.14, FS.15, FS.16, FS.17, FS.18, FS.19, FS.20, FS.21, FS.23, FS.25, FS.26, FS.27, FS.28, FS.29, FS.30, FS.3, FS.32, FS.33, FS.34, FS.36, FS.38, FS.39, FS.40
Elements	**1.1.3.1.1 /:** Root. Serves the web app
	1.1.3.1.2 /api-docs: Api documentation
	1.1.3.1.3 /authorize: Handle authorization concerns
	1.1.3.1.4 **/users:** Handle user concerns
	1.1.3.1.5 **/commercial-users** Handle commercial user concerns
	1.1.3.1.6 **/events:** Handle event concerns
	1.1.3.1.7 **/eventmedia:** Handle event media storage
	1.1.3.1.8 /group-events: Handle group event concerns
	1.1.3.1.9 **/user-report:** Handle user reports
	1.1.3.1.10 **/messages:** Handle messages
	1.1.3.1.11 /bids: Handle bids
	1.1.3.1.12 /payment: Handle payment transactions
	Request.body Contains the Object representing the JSON data passed in the HTTP request.
	JSON On put and post request, A JSON object string passed in the request body.
Referenced by	1.1.3
Viewpoint	Component Diagram

View 1.1.3.x: Facebook Auth Sequence Diagram



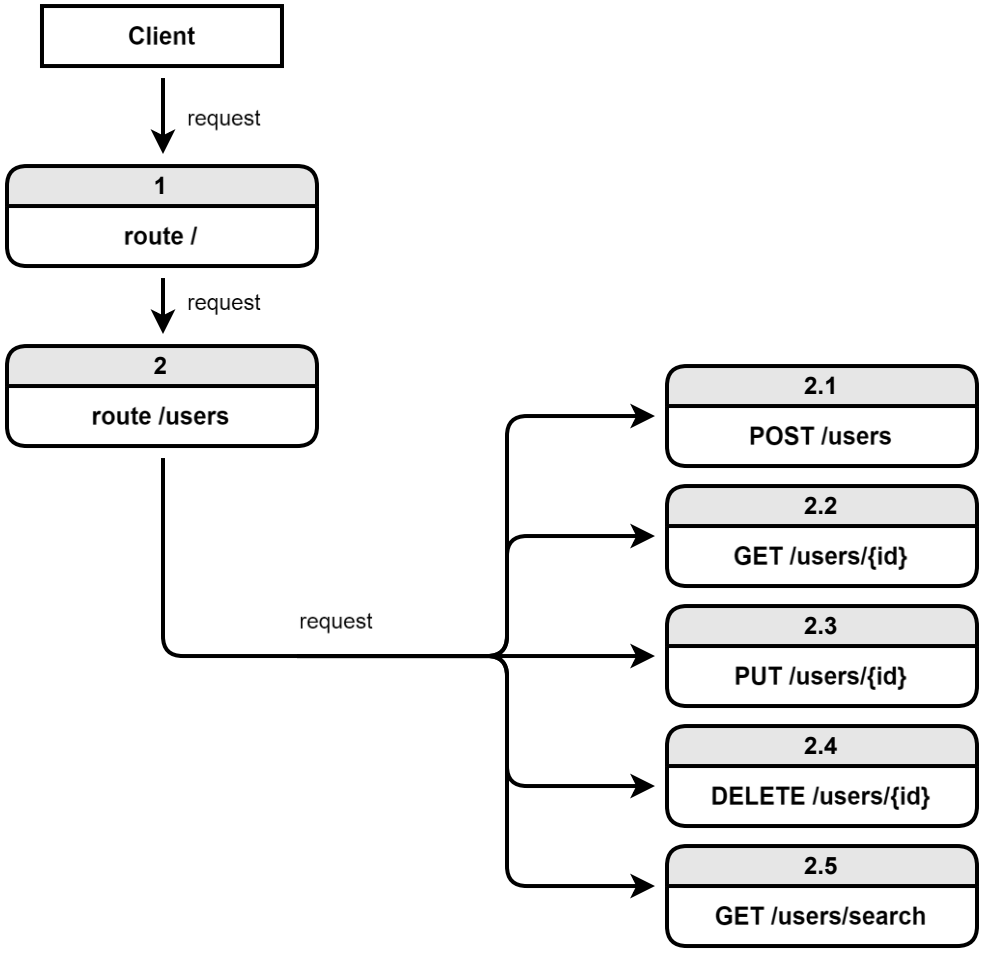
Name	View 1.1.3.x Facebook Auth Sequence Diagram
Purpose	Provide the sequence of events that will need to occur to successfully share an event using the Facebook API.
Description	A diagram showing the sequence of events when sharing an event. This includes the router, controller, validation, logic, database interface and Facebook API.
Requirements	PO.1.1.3 Uvent Server, Facebook API, SAV.2.2, SAV.2.5.2, SAV.4.2
Elements	1.1.3.1 **Router Table:** Module handling all API server routes
	Router- Authentication/Authorization: Beginning of the middleware chain
	1.1.3.2 **Authentication/Authorization:** Module that authenticates users. Can use the Facebook Oauth API.
	Authentication/Authorization-Validation: Express middleware next() call
	1.1.2.1 **Validation:** Module that validates client JSON data
	Validation-Controller: The end of the middleware stack
	1.1.2.2 **Controller:** Applies logic to the request and supplied JSON data
	Controller-Models: Calls various model CRUD operations
	1.1.2.3 **Models:** Model components that interface with the database **1.1.5 Endpoints:** A listing of all API endpoints.
Referenced by	1.1.3
Viewpoint	Sequence Diagram

View 1.1.3.1.3.1 Facebook Auth Data Flow Diagram



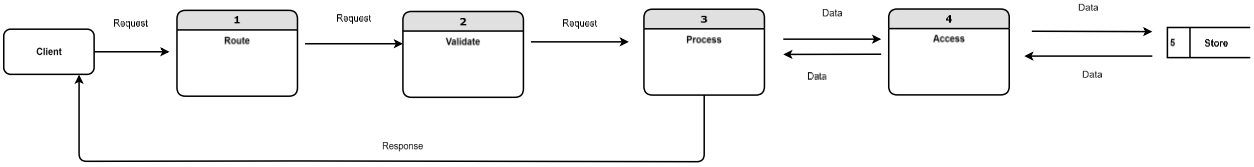
Name	1.1.3.1.3.1 Facebook Auth Data Flow Diagram
Purpose	Provide the flow of data that will need to occur to successfully authenticate a user using the Facebook API.
Description	A diagram showing the flow of data when authenticating with the Facebook API.
Requirements	PO.1.1.3 Uvent Server, Facebook API, SAV.2.2, SAV.2.5.2, SAV.4.2
Elements	1.1.3.1 **Router Table:** Module handling all API server routes **Router- Authentication/Authorization**: Beginning of the middleware chain 1.1.3.2 **Authentication/Authorization:** Module that authenticates users. Can use the Facebook Oauth API. **Authentication/Authorization-Validation:** Express middleware next() call 1.1.2.1 **Validation:** Module that validates client JSON data **Validation-Controller:** The end of the middleware stack 1.1.2.2 **Controller:** Applies logic to the request and supplied JSON data **Controller-Models:** Calls various model CRUD operations 1.1.2.3 **Models:** Model components that interface with the database **1.1.5 Endpoints:** A listing of all API endpoints.
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

(TODO: Change to CD) View 1.1.3.1.4: Sub Route User Data Flow Diagram



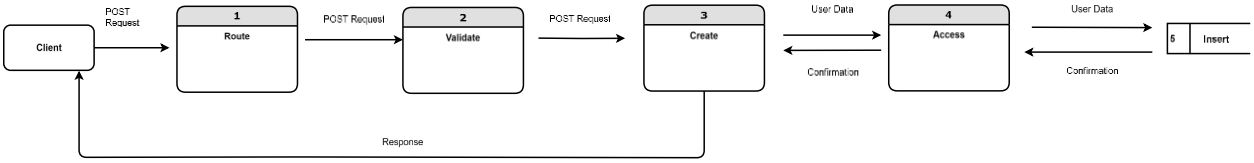
Name	**1.1.3.1.4 Users Sub route**
Purpose	Shows the data flow when a client uses the /users sub route.
Description	When a client uses the /users sub route, data will eventual reach one of the listed endpoints, where it will be processed.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1 **Router:** Root router for the API
	1.1.3.1.4 **Router /users:** Sub router to handle user CRUD
	1.1.3.1.4.1 POST /users: POST handler to create users
	1.1.3.1.4.2 GET /users/{id}: GET handler to retrieve users
	1.1.3.1.4.3 PUT /users/{id}: PUT handler to update users
	1.1.3.1.4.4 DELETE /users/{id}: DELETE handler to delete users
	1.1.3.1.4.5 GET /users/search: GET handler to search for users
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.5: Commercial Users Subroute



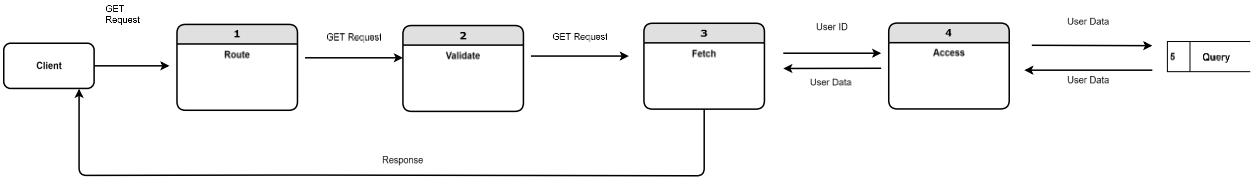
Name	1.1.3.1.5: Commercial Users Subroute DFD
Purpose	Provide the flow of data for handling commercial user operations via various HTTP requests.
Description	System Operations: Handle Commercial User Operations
Requirements	PO.1.7 System Operations: Handle Commercial User Account
Elements	**1.1.3.1 Router: Handles all API server routes.**
	Route: Route request to the Validation.
	1.1.2.2.1 Validation: Module that validates client data.
	Validate: Validate Request to Controller.
	1.1.2 Controller: Applies logic to the request and supplied data.
	Process: Processes the request and interacts with Validation and Model.
	1.1.2.2.3 Model: Model components that interface with the database.
	Access: Access the commercial user account data in the database.
	Database: Stores and retrieves commercial user account data.
	Store: Stores Request data and responds with the status of the database.
Referenced by	PO. 1.1.3 Uvent Server
Viewpoint	Data Flow Diagram

View 1.1.3.1.5.1: Commercial Users POST (create)



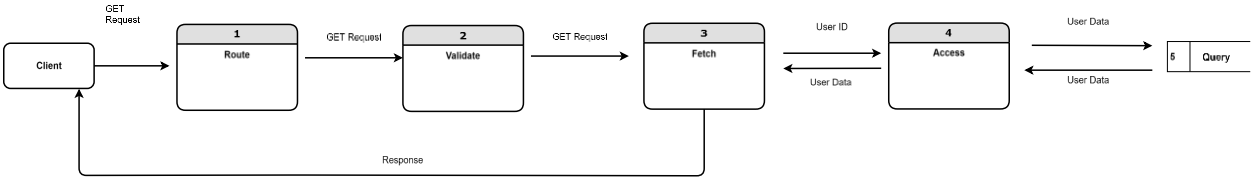
Name	1.1.3.1.5.1: Commercial Users POST (create) DFD
Purpose	Provide the flow of data for creating a commercial user account via a POST request.
Description	System Operations: POST Commercial User Account
Requirements	PO.1.7 System Operations: POST Commercial User Account
Elements	**1.1.3.1 Router: Handles all API server routes.**
	Route: Route request to the Validation.
	1.1.2.2.1 Validation: Module that validates client data.
	Validate: Validate Request to Controller.
	1.1.2 Controller: Applies logic to the request and supplied data.
	Create: Create the user data and interacts with Validation and Model.
	1.1.2.2.3 Model: Model components that interface with the database.
	Access: Access the commercial user account data in the database.
	Database: Stores and retrieves commercial user account data.
	Insert: Insert user data and responds with the confirmation.
Referenced by	PO. 1.1.3 Uvent Server
Viewpoint	Data Flow Diagram

View 1.1.3.1.5.2: Commercial Users GET (read)



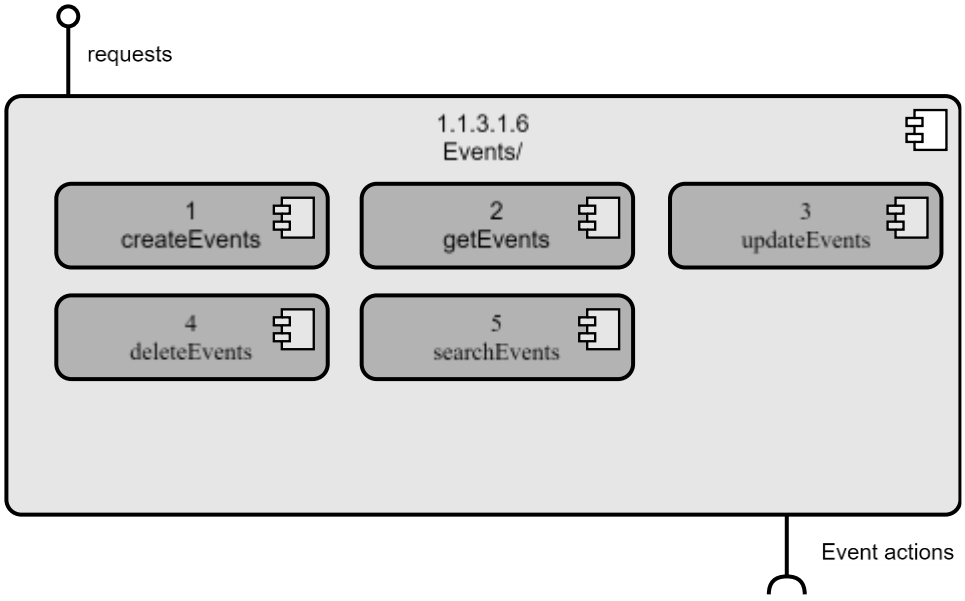
Name	1.1.3.1.5.2: Commercial Users GET (read) DFD
Purpose	Provide the flow of data for creating a commercial user account via a GET (read) request.
Description	System Operations: GET (read) Commercial User Account
Requirements	PO.1.7 System Operations: GET (read) Commercial User Account
Elements	**1.1.3.1 Router: Handles all API server routes.**
	Route: Route request to the Validation.
	1.1.2.2.1 Validation: Module that validates client data.
	Validate: Validate GET Request to Controller.
	1.1.2 Controller: Applies logic to the request and supplied data.
	Fetch: Fetch the user Id and interacts with Validation and Model.
	1.1.2.2.3 Model: Model components that interface with the database.
	Access: Access the commercial user account data in the database.
	Database: Stores and retrieves commercial user account data.
	Query: Query and responds with the user data.
Referenced by	PO. 1.1.3 Uvent Server
Viewpoint	Data Flow Diagram

View 1.1.3.1.5.5: Commercial Users GET (search)



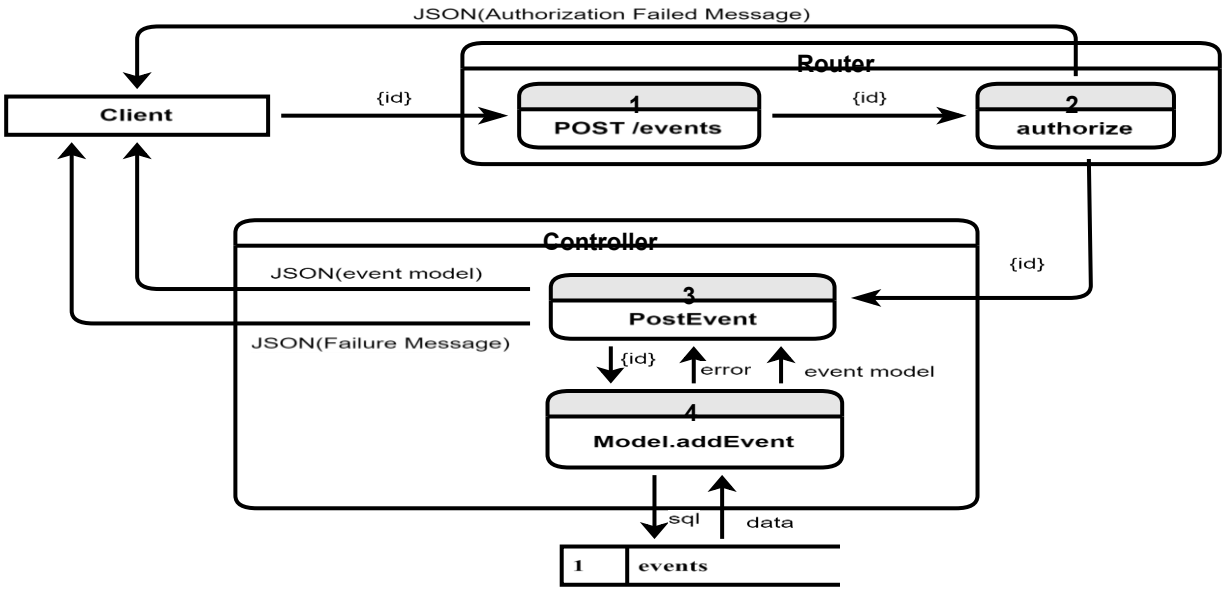
Name	1.1.3.1.5.5: Commercial Users GET (search) DFD
Purpose	Provide the flow of data for creating a commercial user account via a GET (search) request.
Description	System Operations: GET (search) Commercial User Account
Requirements	PO.1.7 System Operations: GET (search) Commercial User Account
Elements	**1.1.3.1 Router : Handles all API server routes.**
	Route: Route search request to the Validation.
	1.1.2.2.1 Validation: Module that validates client data.
	Validate: Validate Search Request to Controller.
	1.1.2 Controller: Applies logic to the request and supplied data.
	Search Users: Search user Id with search criteria and interacts with Validation and Model.
	1.1.2.2.3 Model: Model components that interface with the database.
	Access: Access the commercial user search results in the database.
	Database: Stores and retrieves commercial user account data.
	Query: Query and responds with the user data.
Referenced by	PO. 1.1.3 Uvent Server
Viewpoint	Data Flow Diagram

View 1.1.3.1.6 Event Endpoints



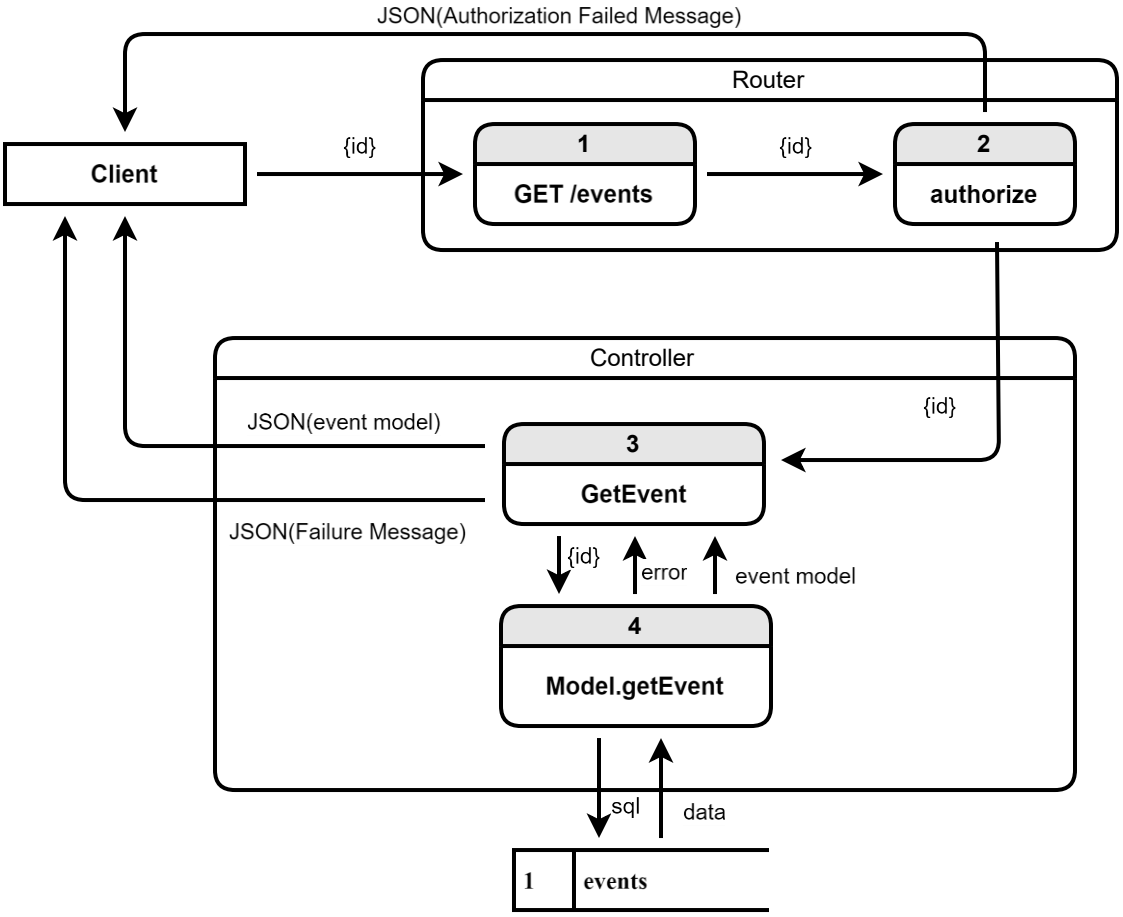
Name	1.1.3.1.6 Events Endpoints
Purpose	To show the messages routes and endpoints through the Uevents application.
Description	When a client sends an events request, this view lists the different routes that the request will take based on the URL and POST/GET/PUT reqests.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**REQUEST** a request from the front end.
	1.1.3.1.6 Events, the events router.
	1 Route to create an event.
	2 Route to get events.
	3 Route to update an event.
	4 Route to delete an event.
	5 Route to search for an event.
	Events Actions, the various functions and actions for each route.
Referenced by	1.1.3.1
Viewpoint	Component Diagram

View 1.1.3.1.6.1: Events POST (create)



Name	1.1.3.1.6.1 Events POST(create)
Purpose	To show how the data flows through the backend when the user requests to make an event on Uevents.
Description	When a client sends a POST request to /group-events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with group event data will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request form the front end
	1.1.3.1.8.1 Router / Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Event addEvent
	1.1.2.3.X Model Event
	Client-1, 1-1, 2-1, 3-1 The id of the event being added.
	2-2 JSON return to client if authorization fails.
	3-2 returns the event data as a JSON object.
	3-3 returns a failure message as a JSON object.
	4-1 SQL request to get the event data from the database
	4-2 returns an error from the model
	4-3 event model returns all the data to be added to the JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

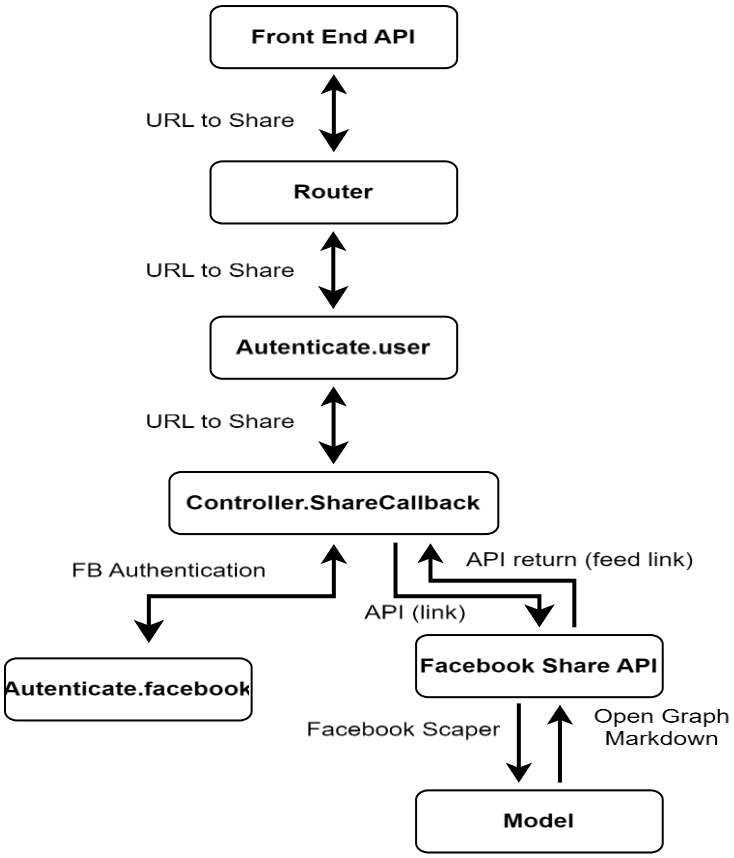
View 1.1.3.1.6.2: Events Get(read)



Name	1.1.3.1.6.2 Events Get(read)
Purpose	To show how the data flows through the backend when the user requests an event on Uevents.
Description	When a client sends a GET request to /group-events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data will be queried from the database and a JSON response with group event data will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request form the front end
	1.1.3.1.8.1 Router / Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Event get Event
	1.1.2.3.X Model Event
	Client-1, 1-1, 2-1, 3-1 The id of the event being requested.
	2-2 JSON return to client if authorization fails.
	3-2 returns the event data as a JSON object.
	3-3 returns a failure message as a JSON object.

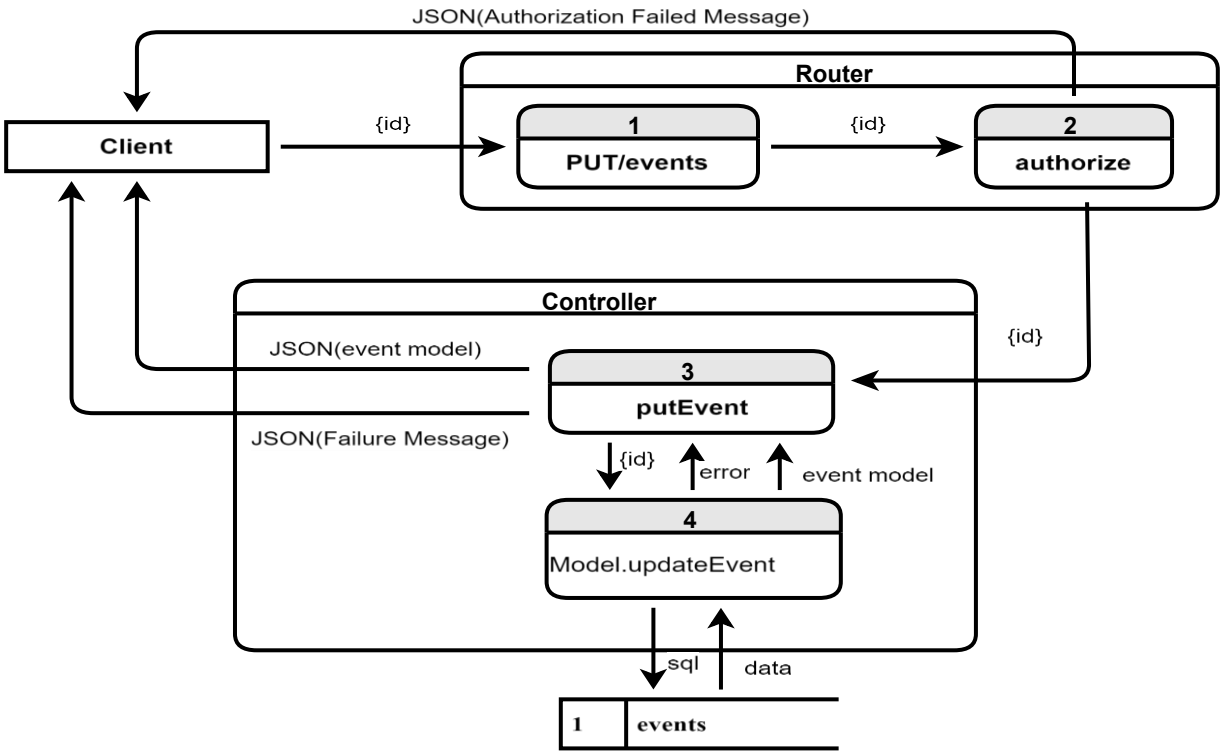
	4-1 SQL request to get the event data from the database
	4-2 returns an error from the model
	4-3 event model returns all the data to be added to the JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.6.2: Facebook Sharing API Structure Chart



Name	1.1.3.1.6.2 Facebook Sharing Structure Chart
Purpose	Provide the call order of structures for the Facebook sharing API.
Description	Provide the call order of structures for the Facebook sharing API.
Requirements	PO.1.1.3 Uvent Server , Facebook API , SAV.2.2, SAV.2.5.2, SAV.4.2
Elements	1.1.3.1 **Router Table:** Module handling all API server routes **Router- Authentication/Authorization**: Beginning of the middleware chain 1.1.3.2 **Authentication/Authorization:** Module that authenticates users. Can use the Facebook Oauth API. **Authentication/Authorization-Validation:** Express middleware next() call 1.1.2.1 **Validation:** Module that validates client JSON data **Validation-Controller:** The end of the middleware stack 1.1.2.2 **Controller:** Applies logic to the request and supplied JSON data **Controller-Models:** Calls various model CRUD operations 1.1.2.3 **Models:** Model components that interface with the database **1.1.5 Endpoints:** A listing of all API endpoints.
Referenced by	1.1.3.1
Viewpoint	Structure chart

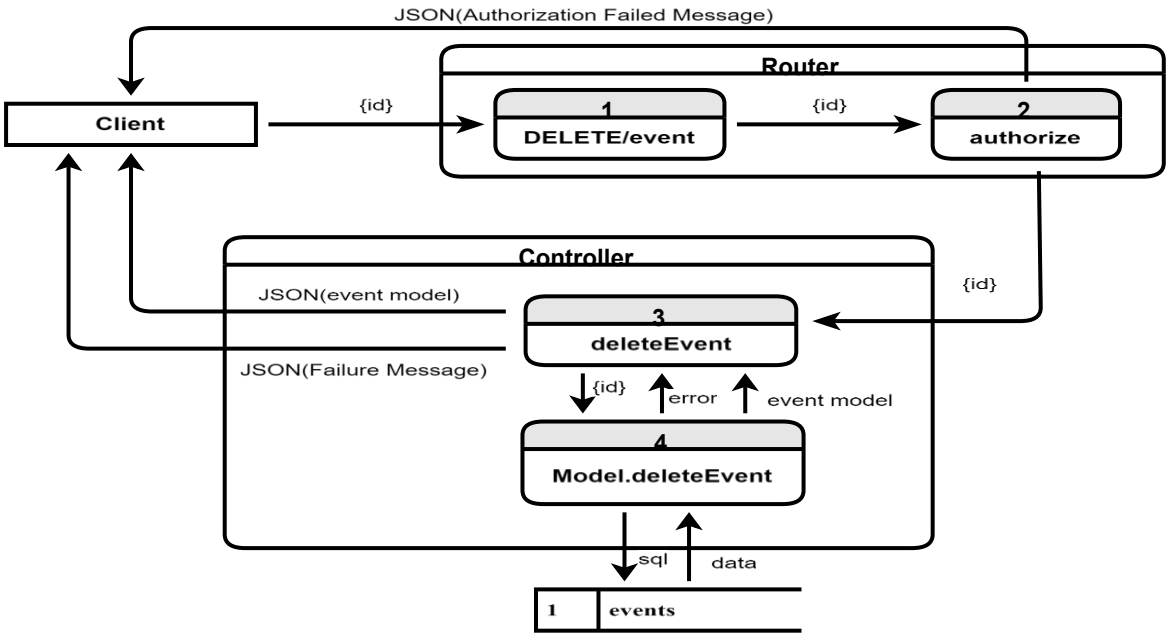
View 1.1.3.1.6.3: Events PUT(update)



Name	1.1.3.1.6.3 Events PUT(update)
Purpose	To show how the data flows through the backend when the user requests to make an update to an event on Uevents.
Description	When a client sends a PUT request to /events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data in the database will be updated and a JSON response will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Event updateEvent
	1.1.2.3.X Model Event
	Client-1, 1-1, 2-1, 3-1 The id of the event being updated.
	2-2 JSON return to client if authorization fails.
	3-2 returns the event data as a JSON object.
	3-3 returns a failure message as a JSON object.
	4-1 SQL request to update the event data in the database
	4-2 returns an error from the model
	4-3 event model returns a summary of the event update as a JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query

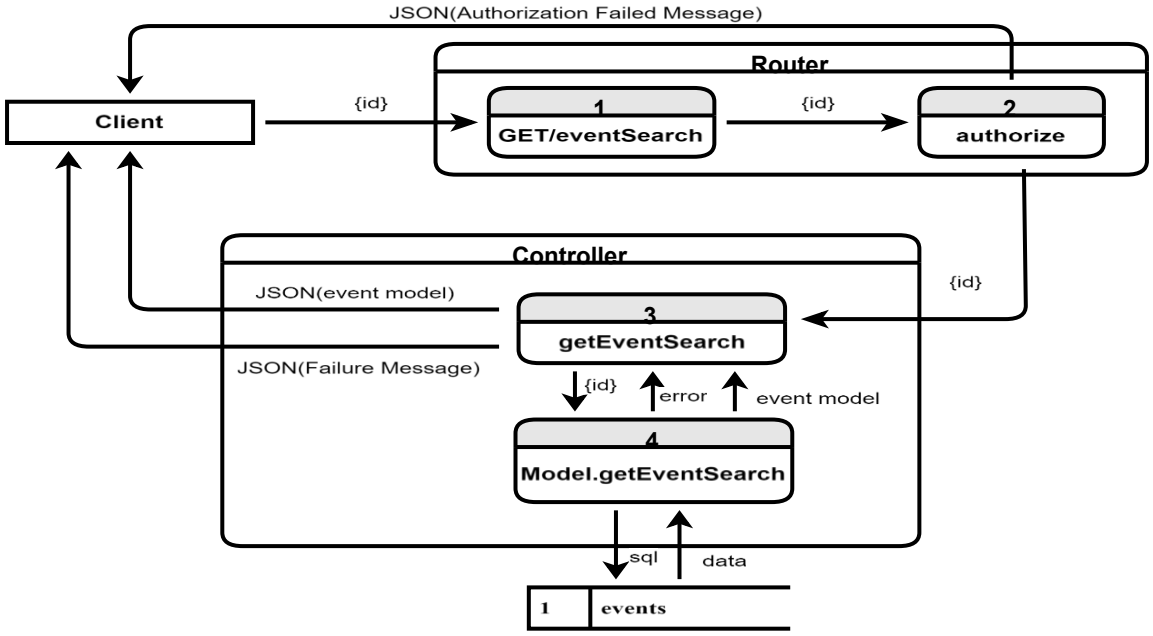
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.6.4: Events DELETE(delete)



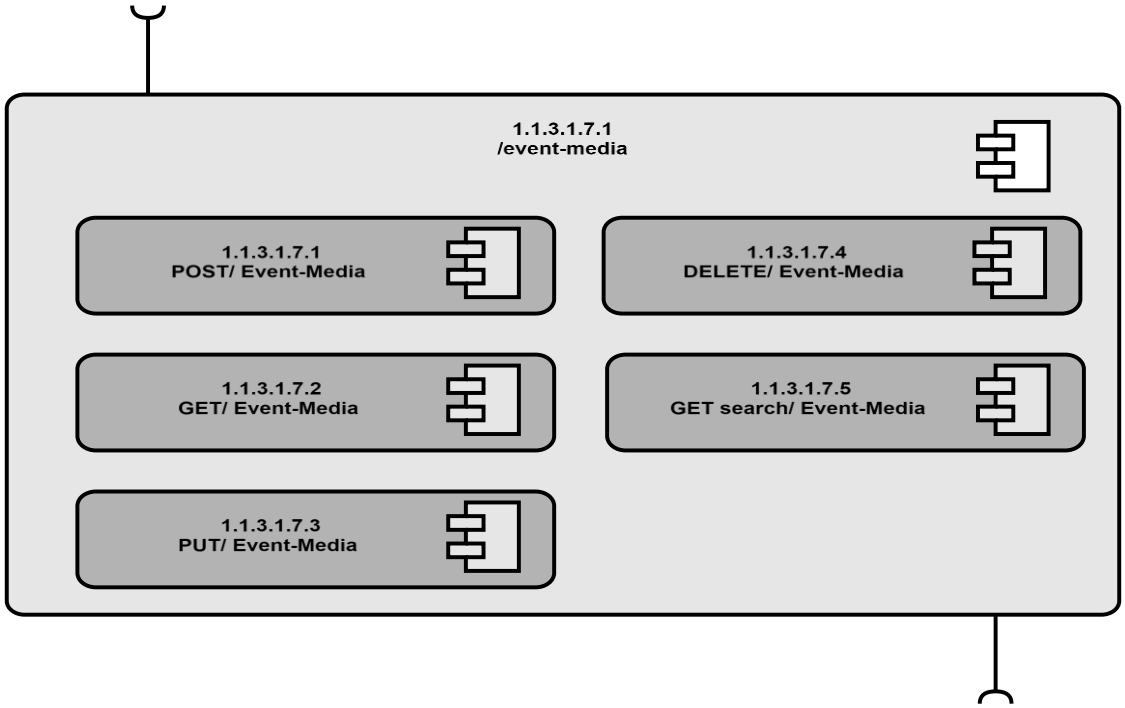
Name	1.1.3.1.6.4 Events DELETE(delete)
Purpose	To show how the data flows through the backend when the user requests to delete an event on Uevents.
Description	When a client sends a DELETE request to /events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the event will be deleted from the database and a JSON response with a summary of the action will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request form the front end
	1.1.3.1.8.1 Router / Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Event deleteEvent
	1.1.2.3.X Model Event
	Client-1, 1-1, 2-1, 3-1 The id of the event being deleted.
	2-2 JSON return to client if authorization fails.
	3-2 returns the status of the event with a JSON object.
	3-3 returns a failure message as a JSON object.
	4-1 SQL request to delete the event data from the database
	4-2 returns an error from the model
	4-3 event model returns all the data to be added to the JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.6.5 Events GET(search)



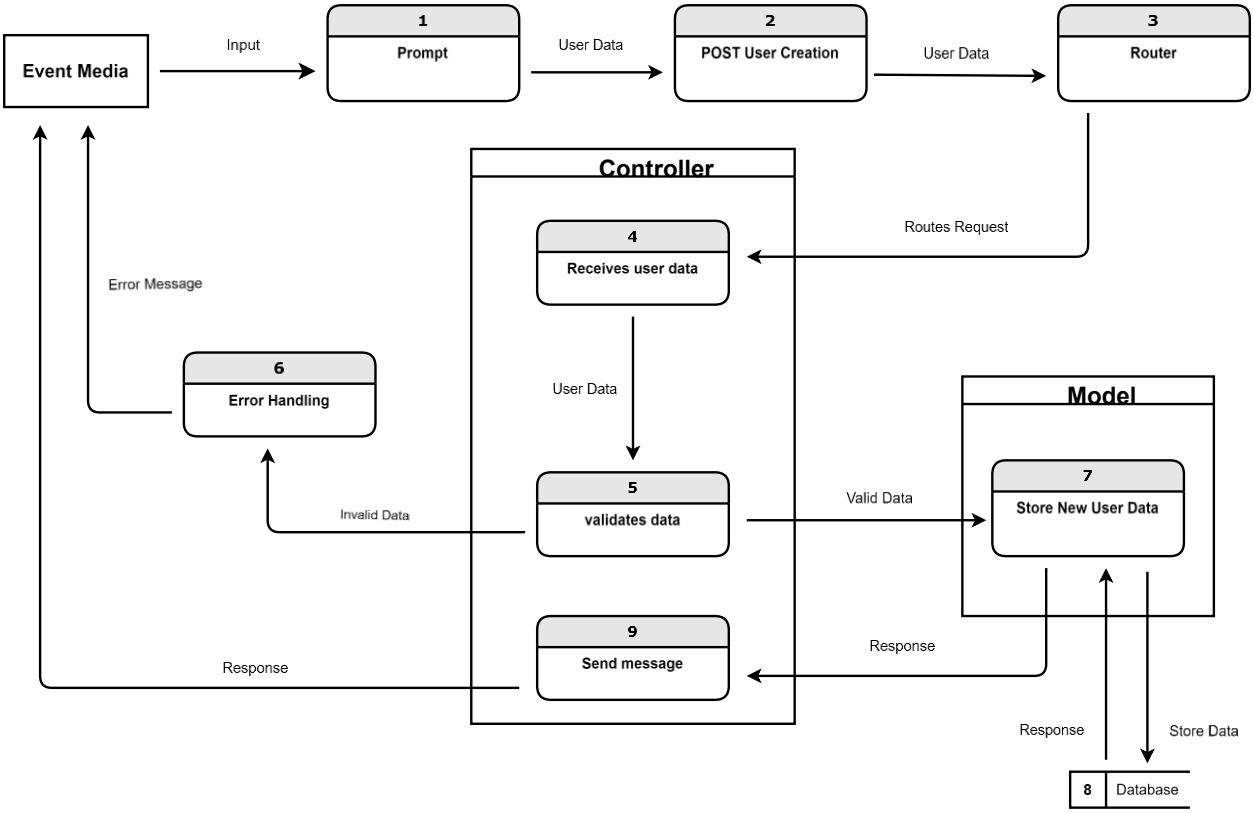
Name	1.1.3.1.6.5 Events POST(create)
Purpose	To show how the data flows through the backend when the user requests an event search on Uevents.
Description	When a client sends a GET search request to /events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with group event data will be returned to the client.
Requirements	FS.7.3, F.S.3 , F.S.11
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Event addEvent
	1.1.2.3.X Model Event
	Client-1, 1-1, 2-1, 3-1, 4-1 The partial id of the event being searched.
	2-2 JSON return to client if authorization fails.
	4-2 returns the event search data as a JSON object.
	4-3 returns a failure message as a JSON object.
	5-1 SQL request to get the event data from the database
	5-2 returns an error from the model
	5-3 event model returns all the matching data as a JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.7 Event Media Sub Route



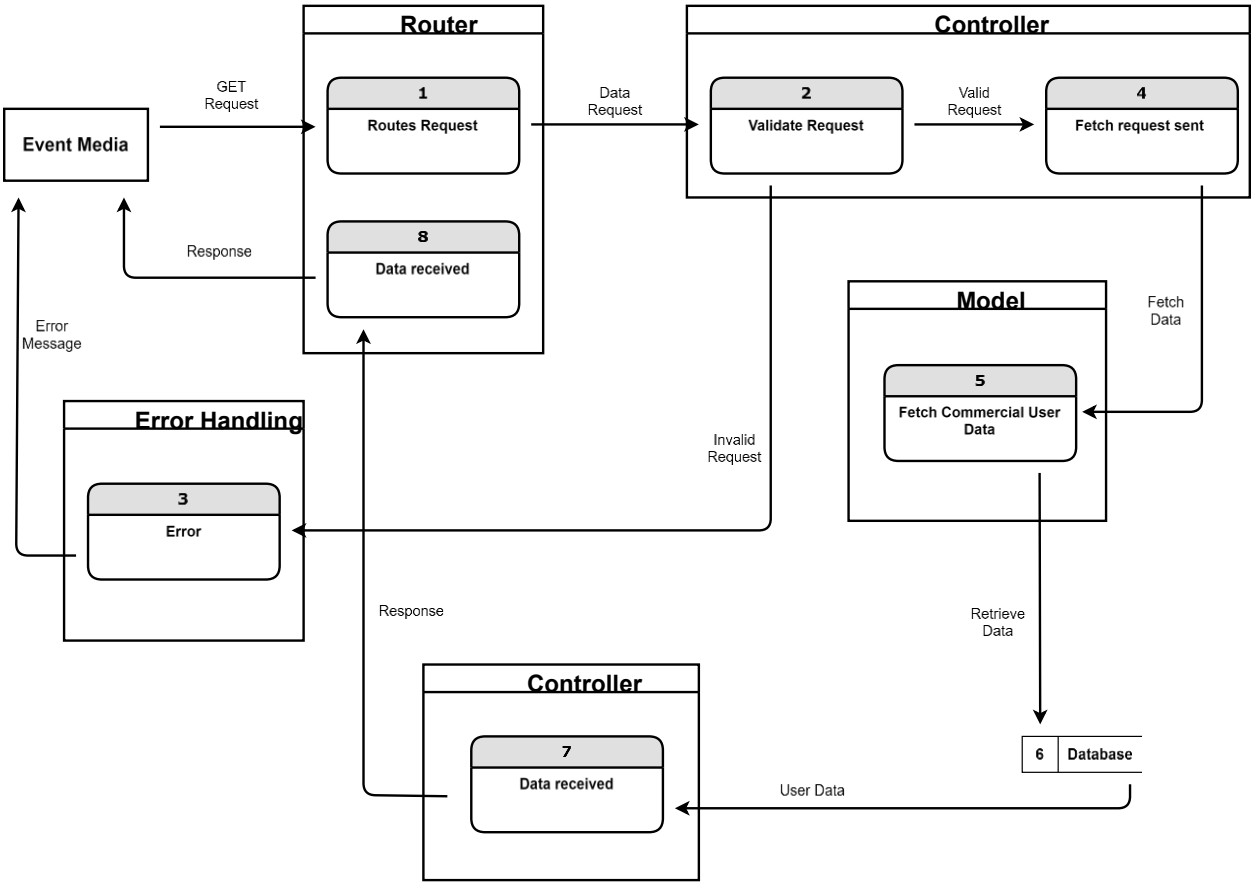
Name	**1.1.3.1.7 Event Media Sub Route**
Purpose	This view is to show all the Endpoints controller for the Event Media
Description	The Event Media endpoints handle the CRUD and search operations.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.7.1 POST/Event-Media is to create a media event
	1.1.3.1.7.2 GET/Event-Media is to retrieve a media event
	1.1.3.1.7.3 PUT/Event-Media is to update a media event
	1.1.3.1.7.4 DELETE/Event-Media is to delete a media event
	1.1.3.1.7.5 GET search/ Event-Media is to search.
Referenced by	1.1.3
Viewpoint	Component Diagram

View 1.1.3.1.7.1 Event Media POST Data Flow Diagram



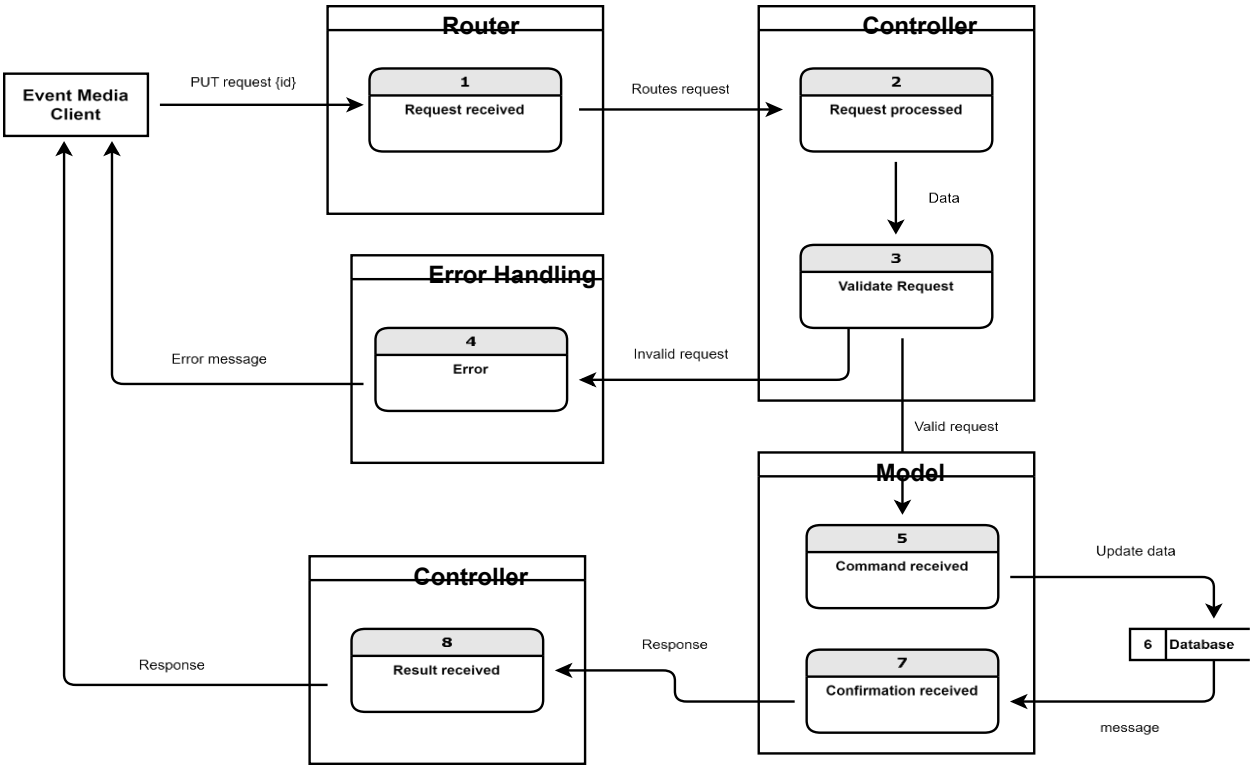
Name	1.1.3.1.7.1 Event Media POST
Purpose	Describe the POST process for Event Media.
Description	This data flow describes the data flow when a user creates an event media through the POST method.
Requirements	PO.2.2.1, PO.3.2, PO.4.6
Elements	1. **User** is prompted to enter their data
	2. **POST** sends user data.
	3. **Router** receives user data then routes the request
	4. **Controller** receives user data
	5**. Controller** validates the data
	6. **Controller** sends error message to user if invalid data was sent
	7. **Model** receives valid data
	8. **Database** receives data to store from **Model** and sends response back
	9. **Controller** receives response from **Model** then send response to user
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.1.7.2 Event Media GET Data Flow Diagram



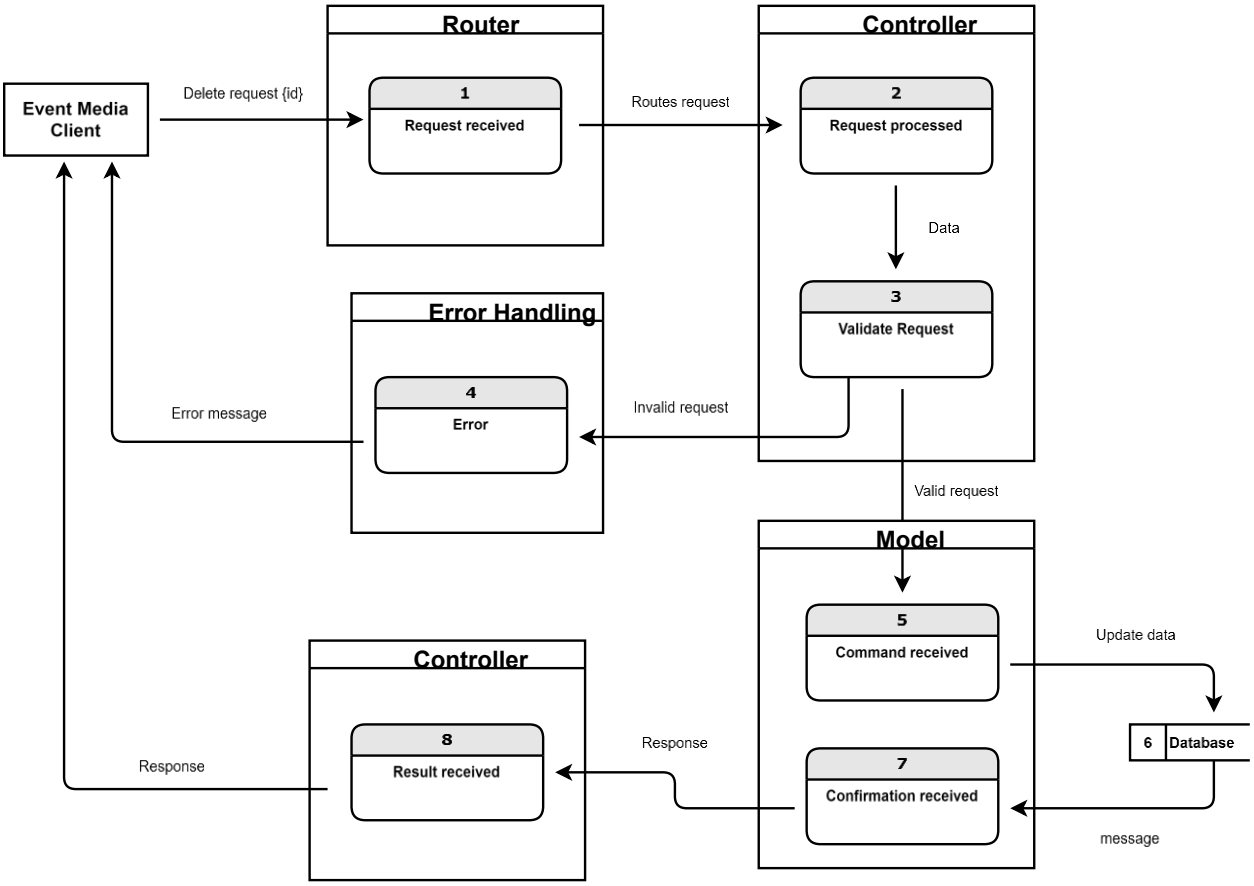
Name	1.1.3.1.7.2 Event Media GET
Purpose	Describe the GET process to retrieve data.
Description	This data flow describes the process to retrieve event media data using the GET method.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1. Client sends request to the Router which then routes the request.
	2. **Controller** receives data request then validates request.
	3. **Controller** send error message to use if request is invalid
	4. If request is valid then Fetch request is sent to **Model**
	5. **Model** retrieves data from database.
	6. Database sends user data to Controller
	7. **Controller** sends response to **Router**
	8. **Router** sends response to client
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.1.7.3 Event Media PUT Data Flow Diagram



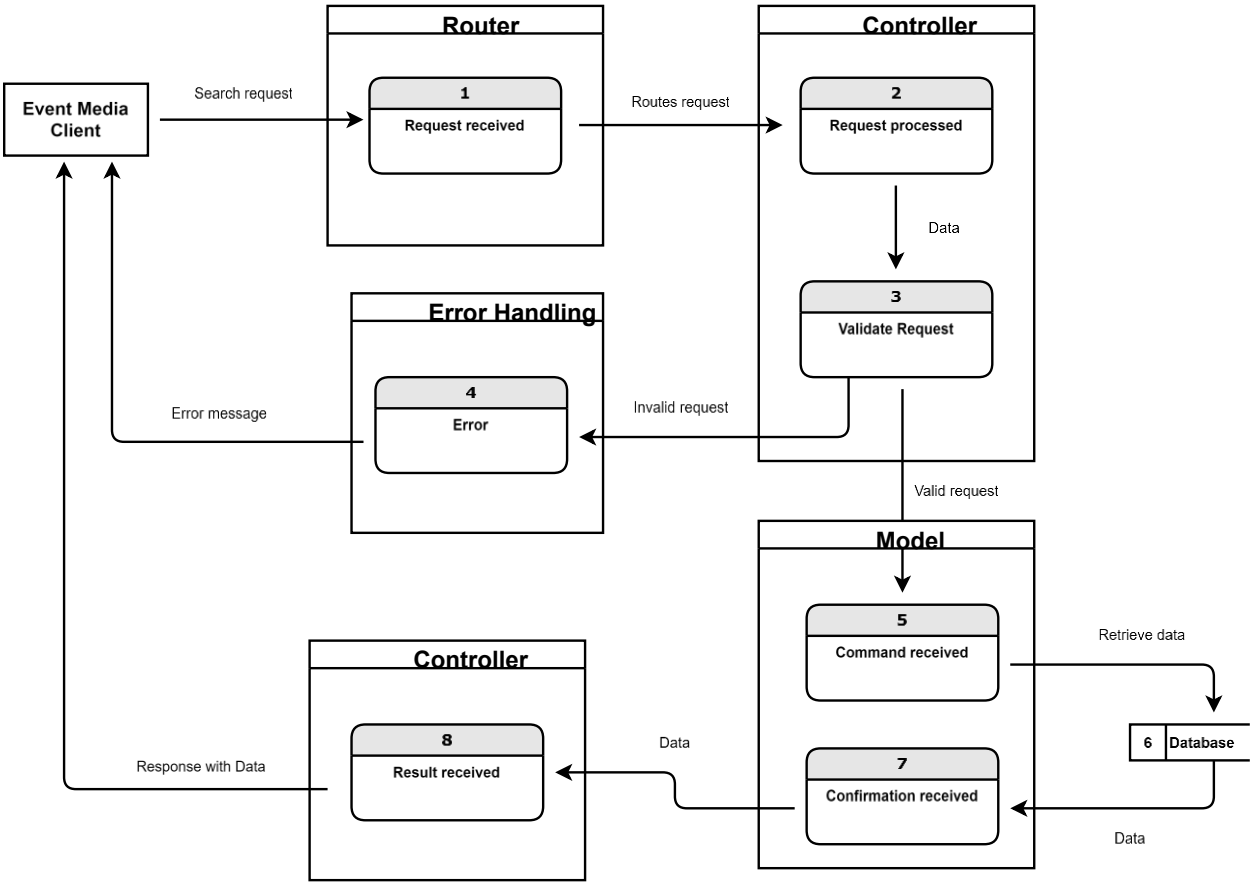
Name	1.1.3.1.7.3 Event Media PUT
Purpose	Show the data flow when an event media client updates data
Description	When a client sends an update request to the system it processes the request to either send an error message or send back a confirmation that the data has been updated after the data has been found and modified.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	Update request is initiated by client
	1. Router receives a PUT request then sends request to the controller
	2. Controller receives and processes the request
	3. Controller validates request
	4. If Controller determines the request is invalid the request is sent to error handling to send error message to client
	5. If request is valid Controller sends the PUT request to the Model to process request and interact with the data base
	6. Database receives the request and modifies the data
	7. Model receives response from database and then sends it to the controller.
	8. Controller receives and prepares the response to send back to the client.
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.1.7.4 Event Media DELETE Data Flow Diagram



Name	1.1.3.1.7.4 Event Media DELETE
Purpose	Show the data flow when an event media client deletes data
Description	When a client sends a delete request to the system it processes the request to either send an error message or send back a confirmation that the data has been deleted after the data has been found and deleted.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	Delete request is initiated by client
	1. Router receives a DELETE request then sends request to the controller
	2. Controller receives and processes the request
	3. Controller validates request
	4. If Controller determines the request is invalid the request is sent to error handling to send error message to client
	5. If request is valid Controller sends the delete request to Model to process request and interact with the data base
	6. Database receives the request and deletes the data
	7. Model receives response from database and then sends it to the controller.
	8. Controller receives and prepares the response to send back to the client.
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

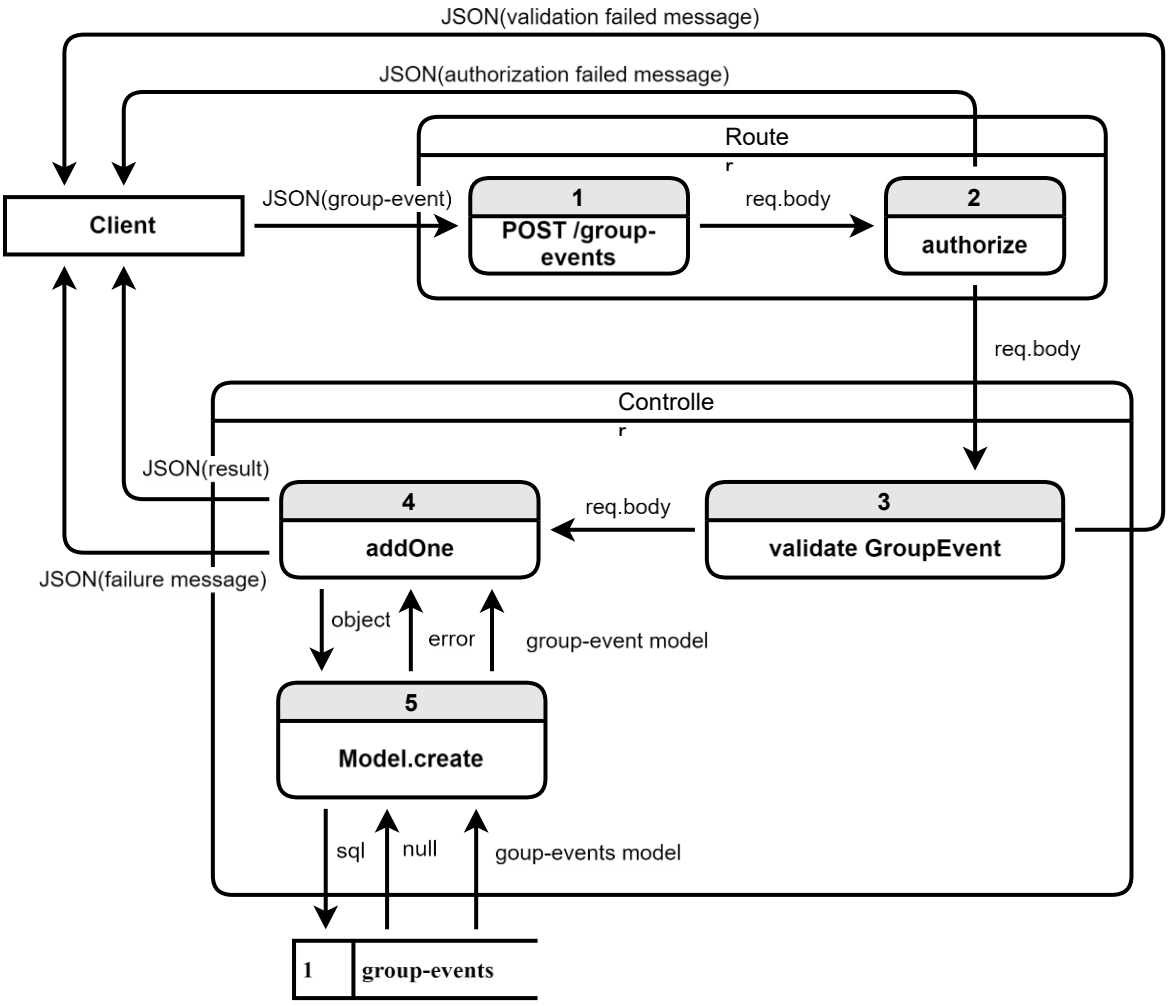
View 1.1.3.1.7.5 Event Media GET (search) Data Flow Diagram



Name	1.1.3.1.7.5 Event Media GET(search)
Purpose	Show the data flow when an event media client searches for data
Description	When a client sends a request to search for data the system processes the request to either send an error message or send back the data requested if found.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	Search request is initiated by client
	1. Router receives a GET request then send request to the controller
	2. Controller receives and processes the request
	3. Controller validates request
	4. If Controller determines the request is invalid the request is sent to error handling to send error message to client
	5. If request is valid Controller send request to Model to process request
	6. Database retrieves data and send it back to the model
	7. Model receives data and send it to the controller.
	8. Controller receives data and prepares the response to send back to the client.
Referenced by	1.1.3

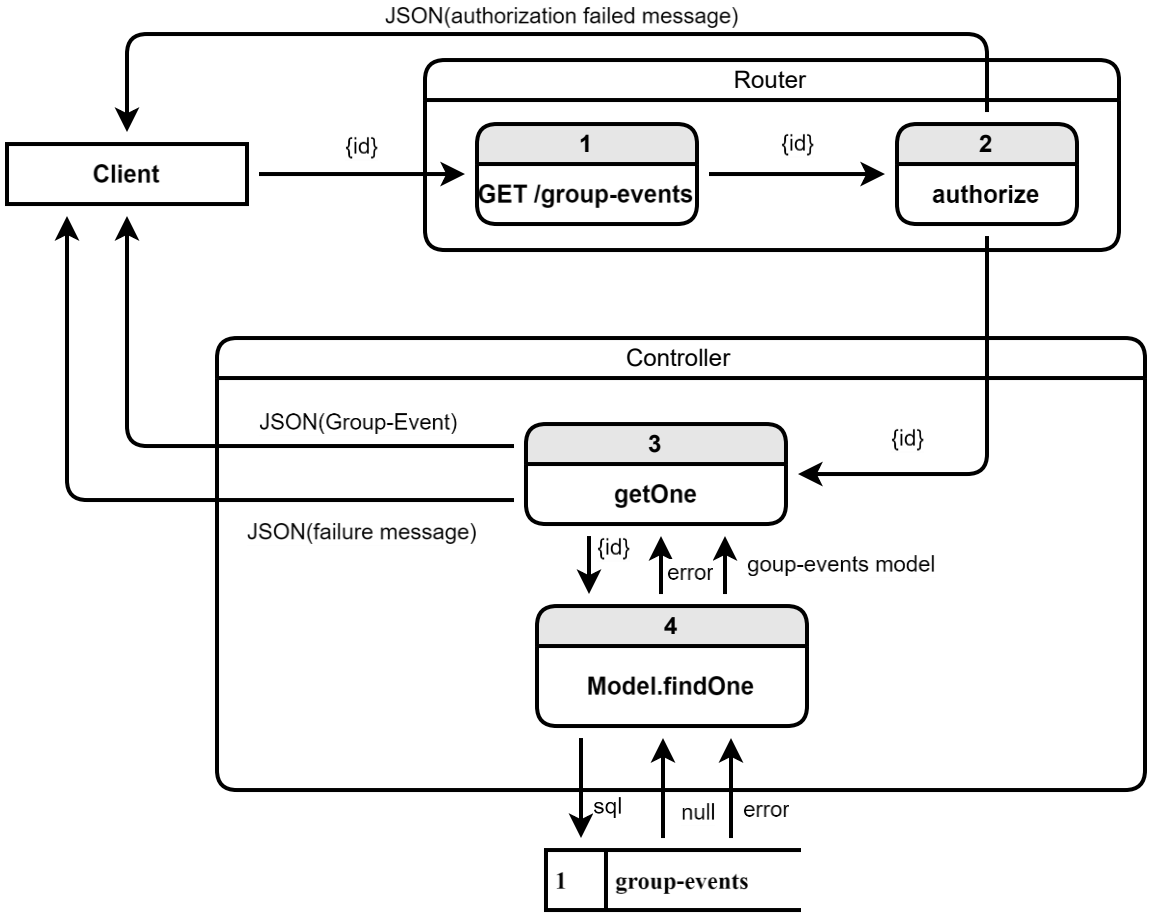
Viewpoint	Data Flow Diagram

View 1.1.3.1.8.1: Group Events POST Data Flow Diagram



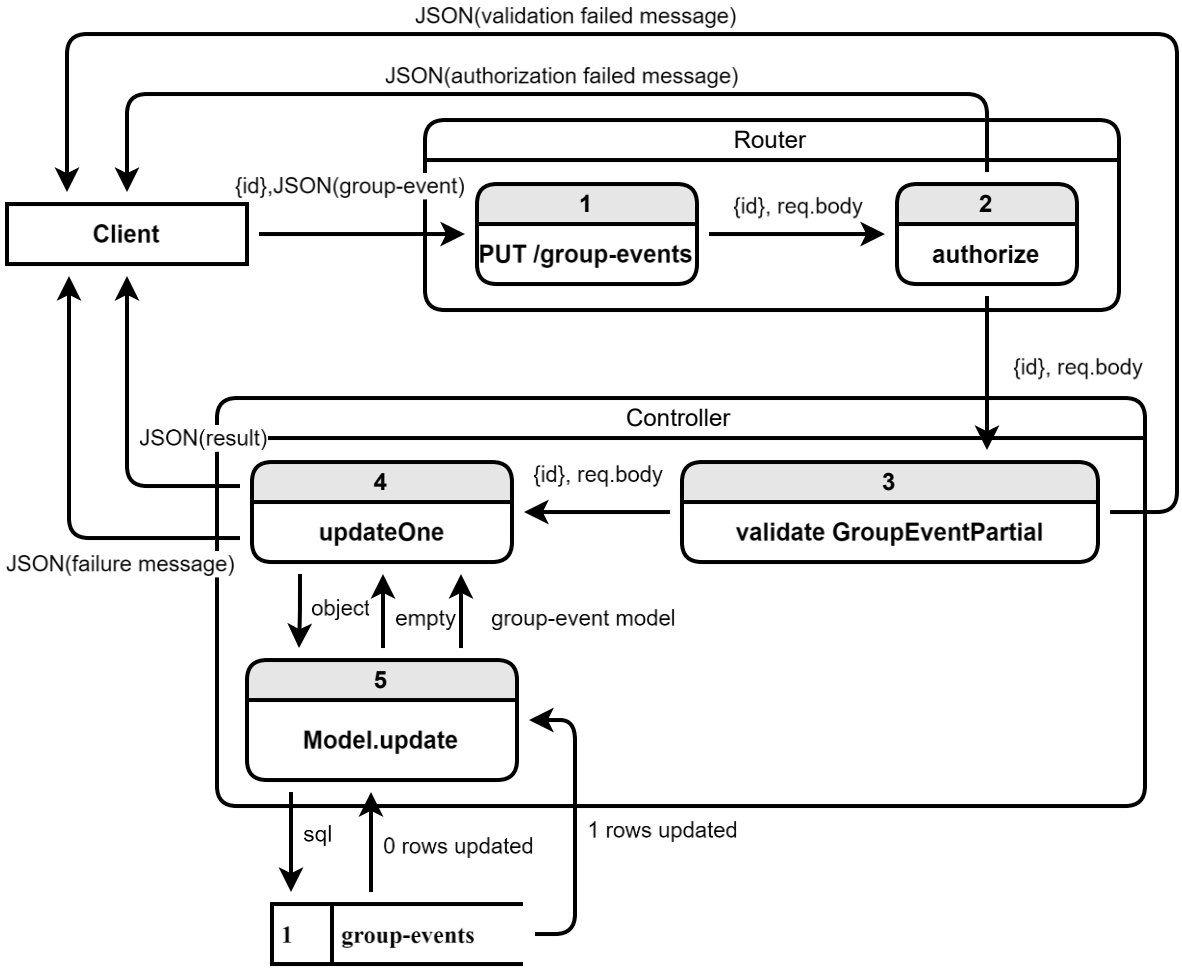
Name	1.1.3.1.8.1 Group Event Post Endpoint
Purpose	Show the flow of data when a client creates a group event
Description	When a client sends a POST request to /group-events, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.8.1 Router /Group Events
	1.1.3.2 Authorize
	1.1.2.1.X.X Validation validateGroupEvent
	1.1.2.2.X.X Controller Group Event addOne
	1.1.2.3.X Model Group Event
	1.1.1.X group-events table
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.8.2 Group Events GET Data Flow Diagram



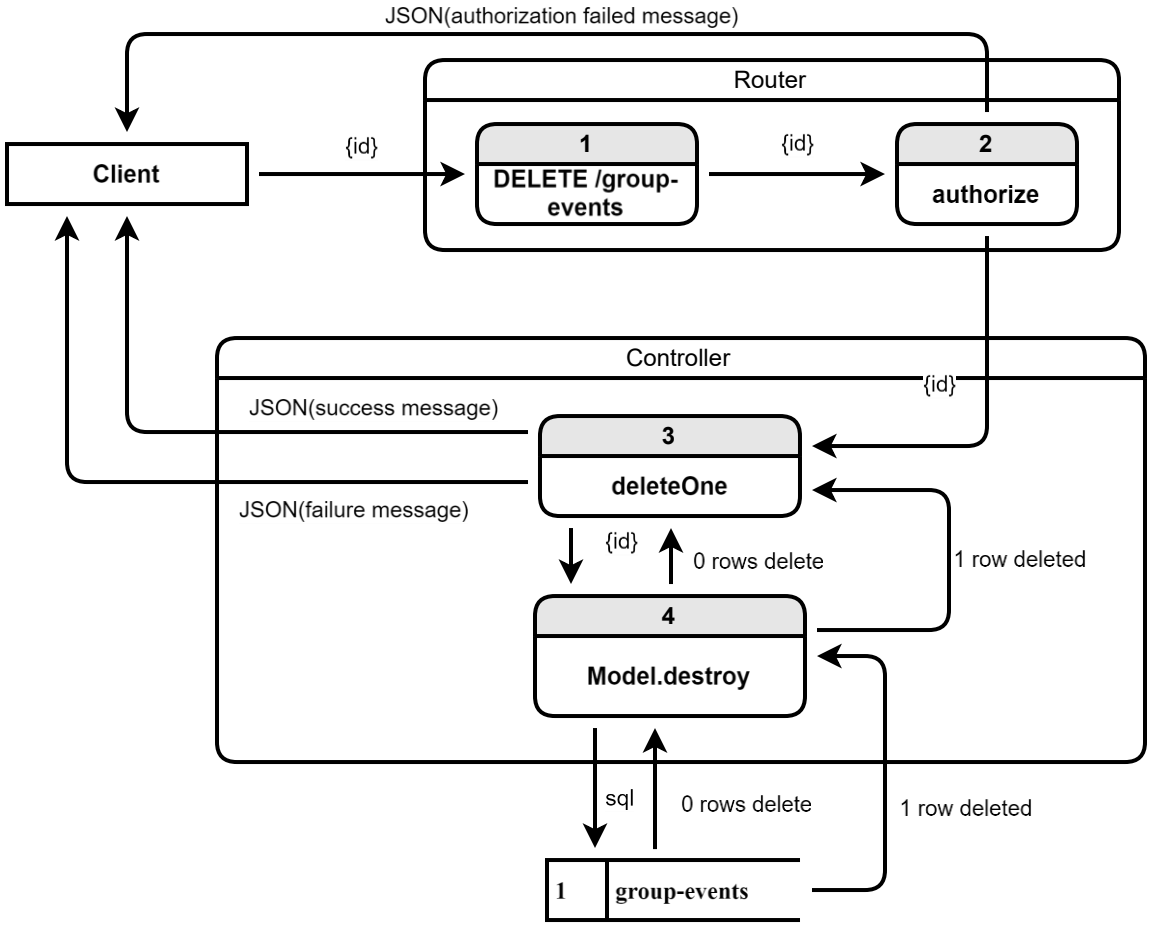
Name	1.1.3.1.8.2 Group Event Get Endpoint
Purpose	Show the flow of data when a client retrieves a group event
Description	When a client sends a GET request to /group-events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with group event data will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.8.1 Router /Group Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Group Event getOne
	1.1.2.3.X Model Group Event
	1.1.1.X group-events table
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.8.3 Group Events PUT Data Flow Diagram



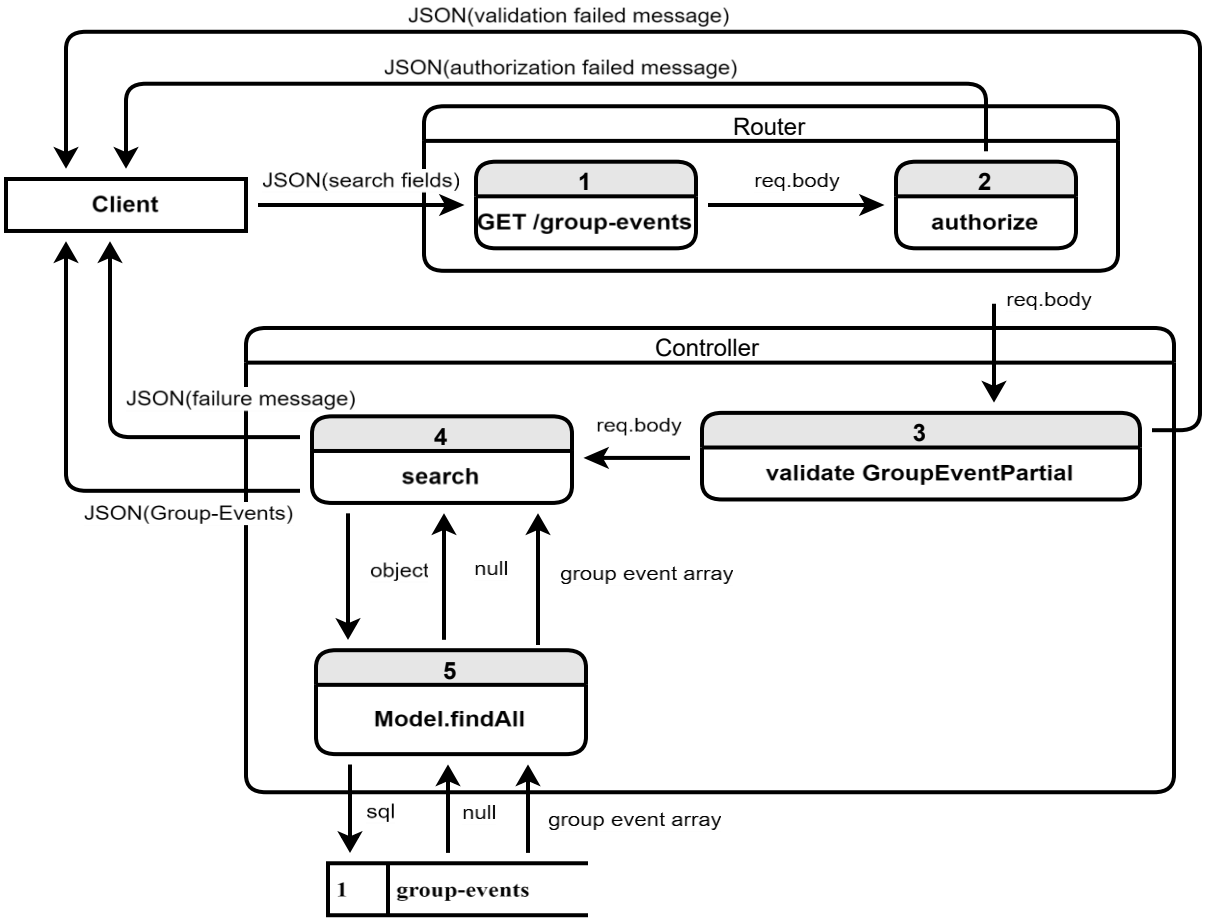
Name	1.1.3.1.8.3 Group Event Put Endpoint
Purpose	Show the flow of data when a client updates a group event
Description	When a client sends a PUT request to /group-events, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will updated in the database and a JSON response will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.8.1 Router /Group Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Group Event updateOne
	1.1.2.3.X Model Group Event
	1.1.1.X group-events table
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.8.4 Group Events DELETE Data Flow Diagram



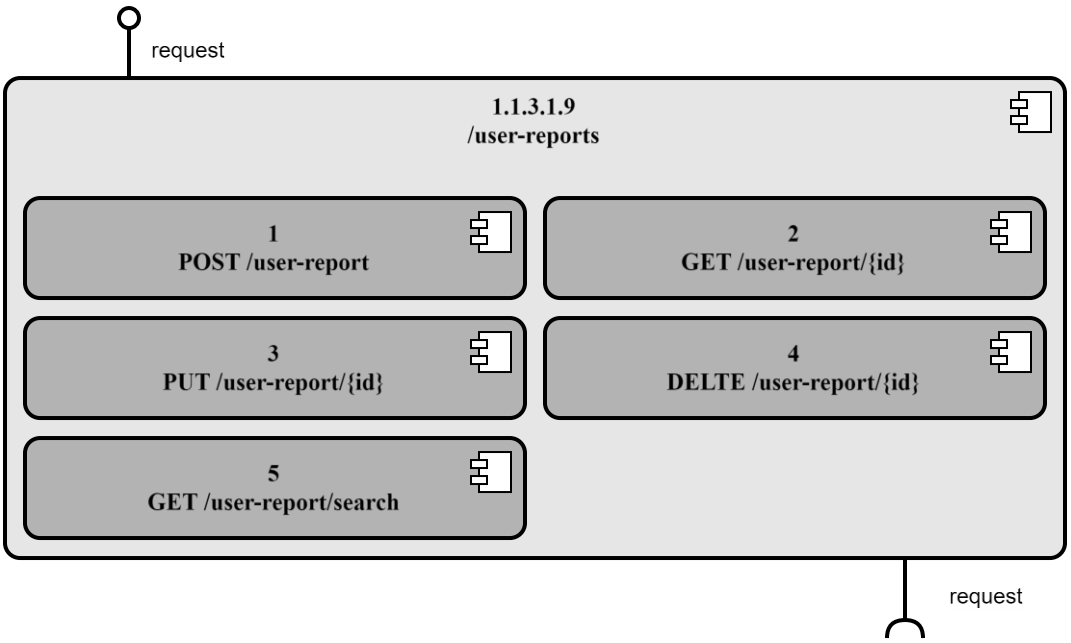
Name	1.1.3.1.8.4 Group Event Delete Endpoint
Purpose	Show the flow of data when a client deletes a group event
Description	When a client sends a DELETE request to /group-events, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data will be removed from the database and a JSON response confirmation will be sent to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.8.1 Router /Group Events
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller Group Event deleteOne
	1.1.2.3.X Model Group Event
	1.1.1.X group-events table
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.8.5 Group Events GET Search Data Flow Diagram



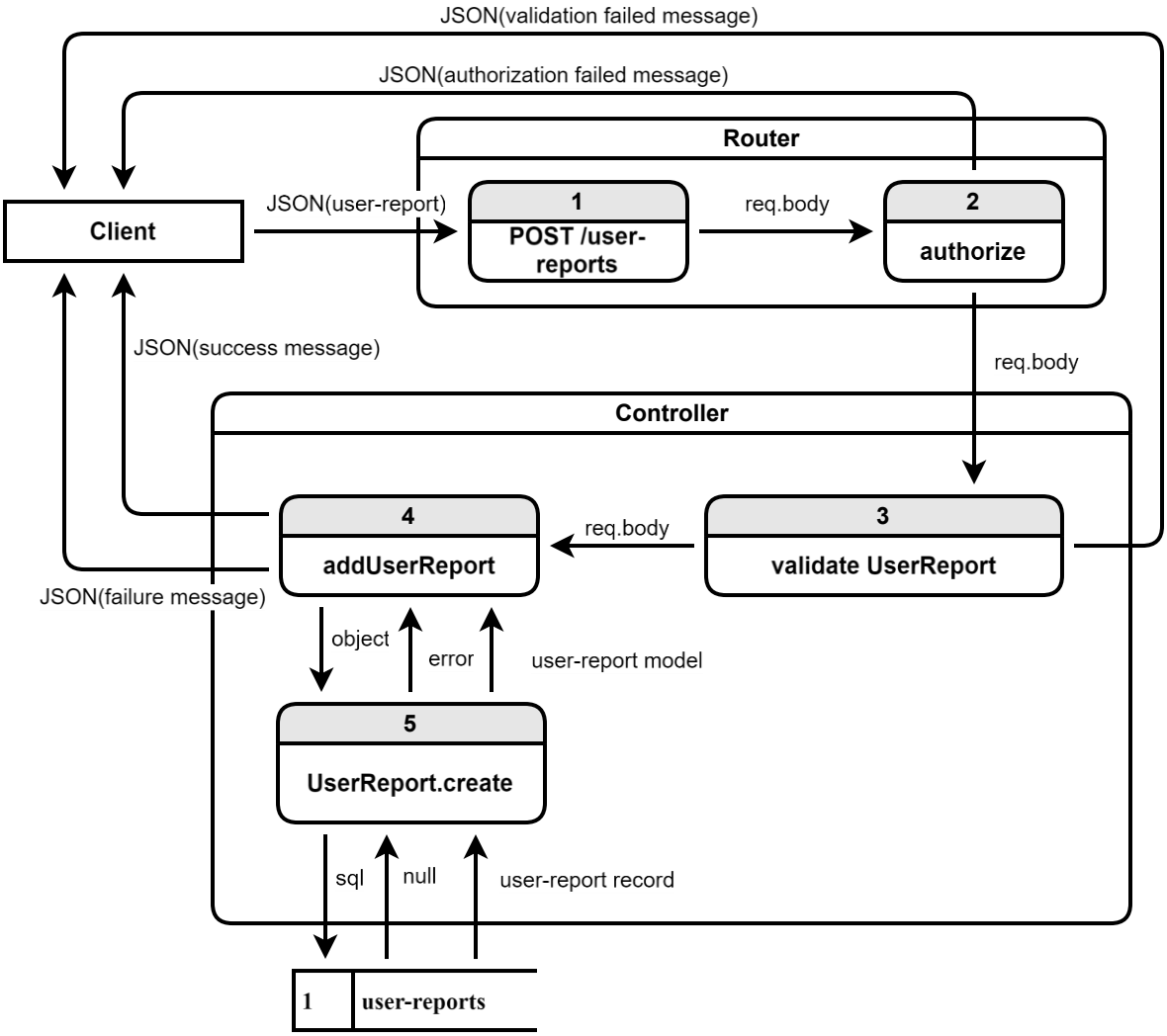
Name	1.1.3.1.8.5 Group Event Get Search Endpoint
Purpose	Show the flow of data when a client searches for a group event
Description	When a client sends a GET request to /group-events, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, that data is used as a field match to retrieve group event data. On a successful query, a JSON response with the data will be returned to the client. On a failed query, the failure message is sent.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	1.1.3.1.8.1 Router /Group Events
	1.1.3.2 Authorize
	1.1.2.1.X.X Validation validateGroupEventPartial
	1.1.2.2.X.X Controller Group Event search
	1.1.2.3.X Model Group Event
	1.1.1.X group-events table
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.9 User Report Sub Route



Name	1.1.3.1.9 User Report Sub Route
Purpose	Shows the endpoints available when a client uses the /user-reports sub route.
Description	When a client uses the /user-reports sub route, data will eventual reach one of the listed endpoints, where it will be processed.
Requirements	LDR.7, LDV.7
Elements	1.1.3.1.9.1 **POST /user-reports:** POST handler to create users
	1.1.3.1.9.2 **GET /user-reports/{id}:** GET handler to retrieve users
	1.1.3.1.9.3 **PUT /user-reports/{id}:** PUT handler to update users
	1.1.3.1.9.4 **DELETE /user-reports/{id}:** DELETE handler to delete user reports
	1.1.3.1.9.5 **GET / group-events /search:** GET handler to search for users
	Request: The standard Express request object. It may contain a body object which here will be a JavaScript object representation of a JSON string passed by the client. It can also contain a param object which contains parameters passed as part of the URL, most often an id field.
Referenced by	1.1.3.1
Viewpoint	Component Diagram

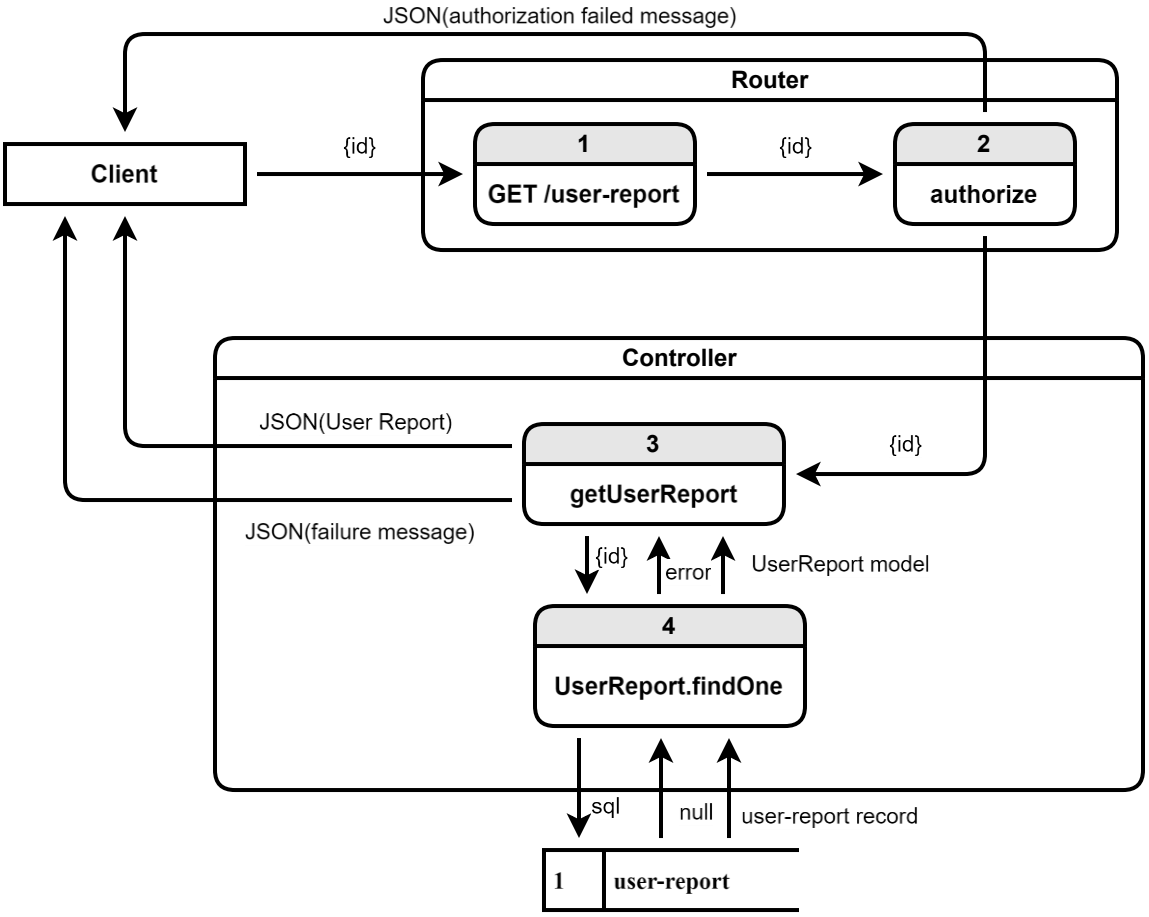
View 1.1.3.1.9.1: User Report POST Data Flow Diagram



Name	1.1.3.1.9.1 User Report Post Endpoint
Purpose	Show the flow of data when a client creates a user report
Description	When a client sends a POST request to /user-report, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response will be returned to the client.
Requirements	LDR.7, LDV.7
Elements	1.1.3.1.9.1 **Router /user-reports:** The route entry point to handle user reports
	1.1.3.2 **Authorize** Checks if the user is authorized to use this resource
	1.1.2.1.6 Validation validateUserReport Middleware that validates the fields of a user report
	1.1.2.2.9.1 Controller User Report addUserReport The controller function that manages inserting the user report in to the database and creating a http response to the client.

	1.1.2.3.9 Model UserReport Sequelize model of user reports
	1.1.1.7 user-report table Database table for user reports
	Client-1 JSON JSON string in the HTTP request body field.
	1-2, 2-3, 3-4, Req The standard Express request object. It may contain a body object which here will be a JavaScript object representation of a JSON string passed by the client. It can also contain a param object which contains parameters passed as part of the URL, most often an id field.
	4-5 object JavaScript object containing the User Report data
	5-user-reports sql Sequelize generated SQL query to create the user report record
	User-reports-user-report record The result of the SQL query, or null if failed.
	1.1.2.3.9 5-4 user-report model An instance of the UserReport model containing the newly created data
	4-client success/fail message A JSON string with a message field stating the API call was successful or not
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

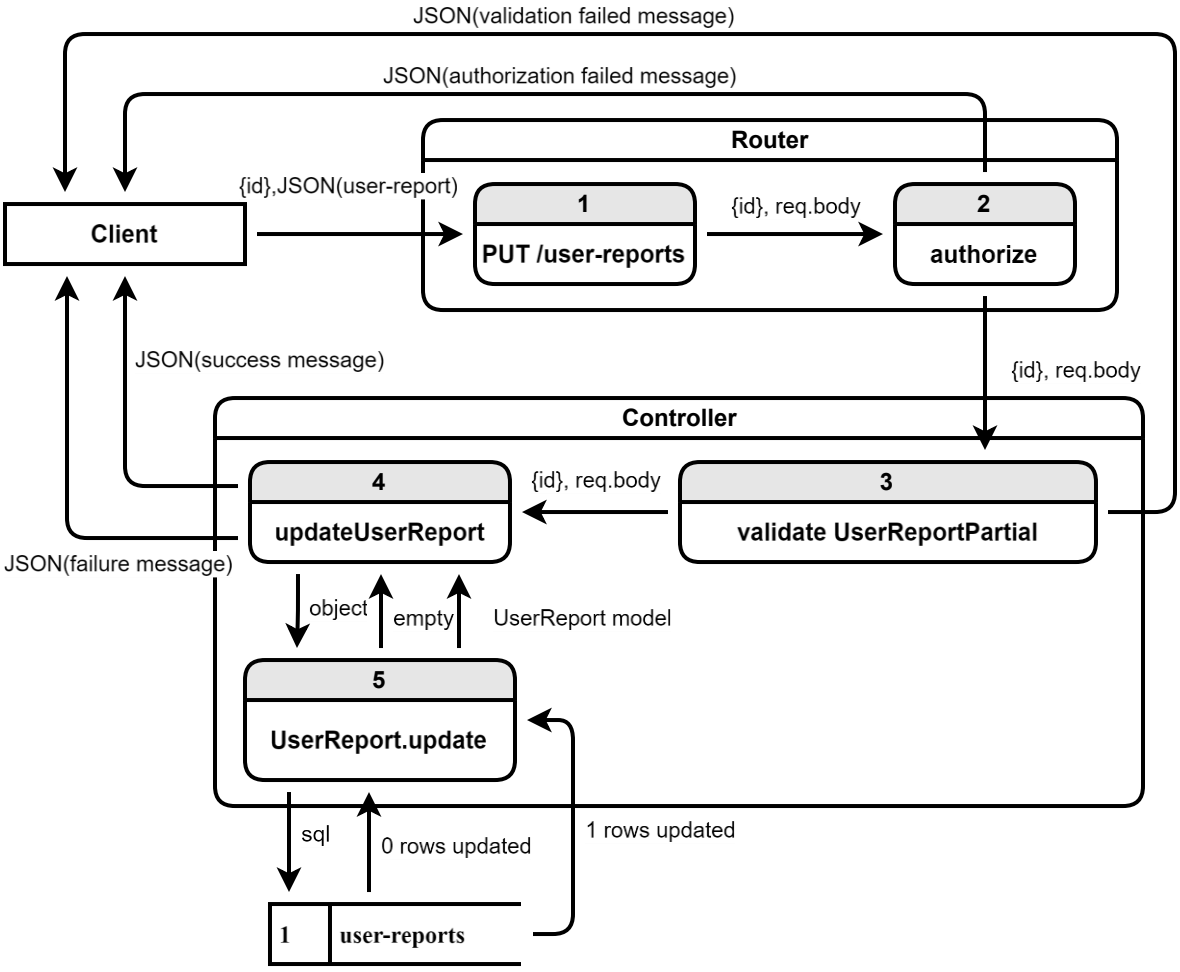
View 1.1.3.1.9.2 User Report Get Endpoint



Name	1.1.3.1.9.2 User Report Get Endpoint
Purpose	Show the flow of data when a client retrieves a user report
Description	When a client sends a GET request to /user-report, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with user report data will be returned to the client.
Requirements	LDR.7, LDV.7
Elements	**1.1.3.1.9.1 Router /user-reports** The route entry point to handle user reports
	1.1.3.2 Authorize Module to authorize users to access this resource
	1.1.2.2.9.1 Controller User Report Event getUserReport
	1.1.2.3.9 Model UserReport Sequelize model of user reports
	1.1.1.7 user-report table Database table for user reports
	Client-1 id ID number parameter for the user report
	1-2, 2-3 id ID number passed through the req.param.id field
	3-4 id ID passed as an integer
	4-user-report sql SQL query to retrieve the user report

	User-report-4 user report record SQL query result containing the user report
	1.1.2.3.9 4-3 UserReport model An instance the the UserReport model containing the desired user report or error.
	3-Client User Report A JSON object string containing the requested user report or a failure message
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

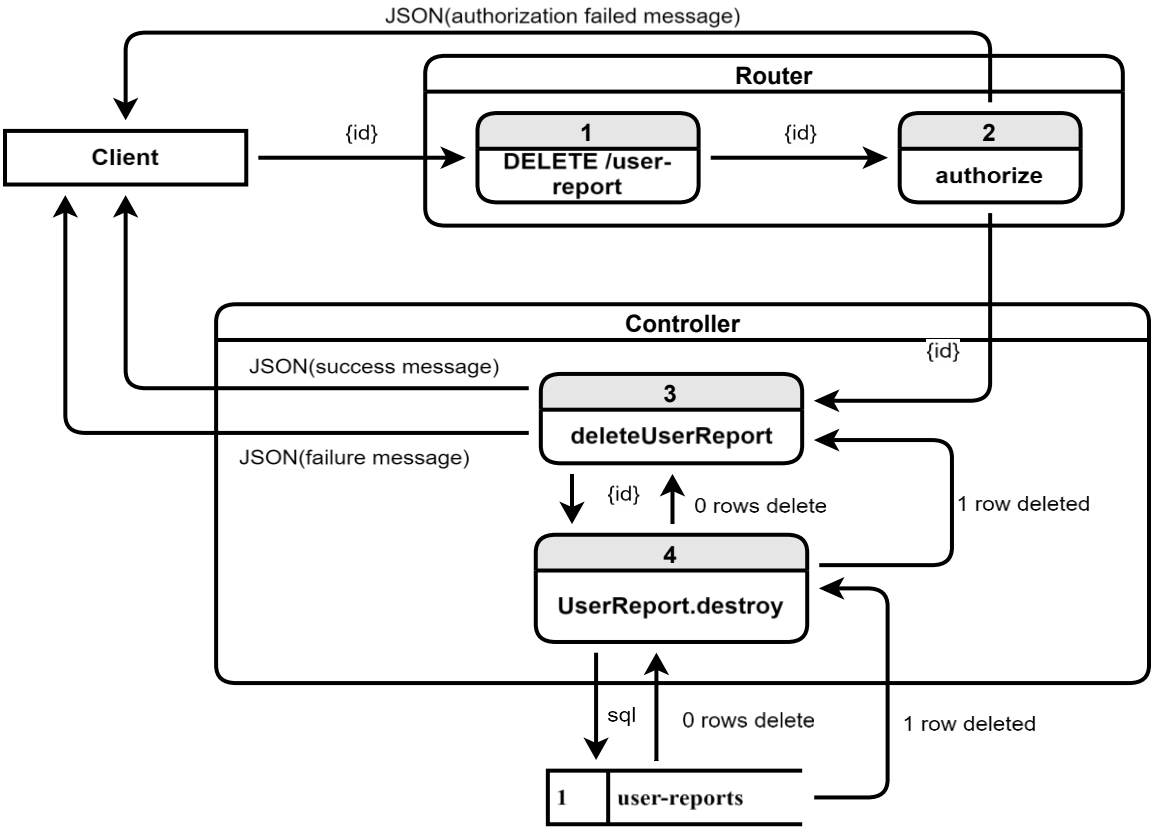
View 1.1.3.1.9.3 User Report Put Endpoint



Name	1.1.3.1.9.3 User Report Put Endpoint
Purpose	Show the flow of data when a client updates a user report
Description	When a client sends a PUT request to /user-report, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will updated in the database and a JSON response will be returned to the client.
Requirements	LDR.7, LDV.7
Elements	**1.1.3.1.9.1 Router /user-reports** The route entry point to handle user reports
	1.1.3.2 Authorize Module to authorize users to access this resource
	1.1.2.2.9.1 Controller UserReport updateUserReport Controller method to update a user report
	1.1.2.3.X Model UserReport Sequelize model of user reports
	1.1.1.7 user-reports table Database table for user reports
	Client-1 {id} JSON id is passed as a parameter with the request url while that updated fields are in the body as a JSON document

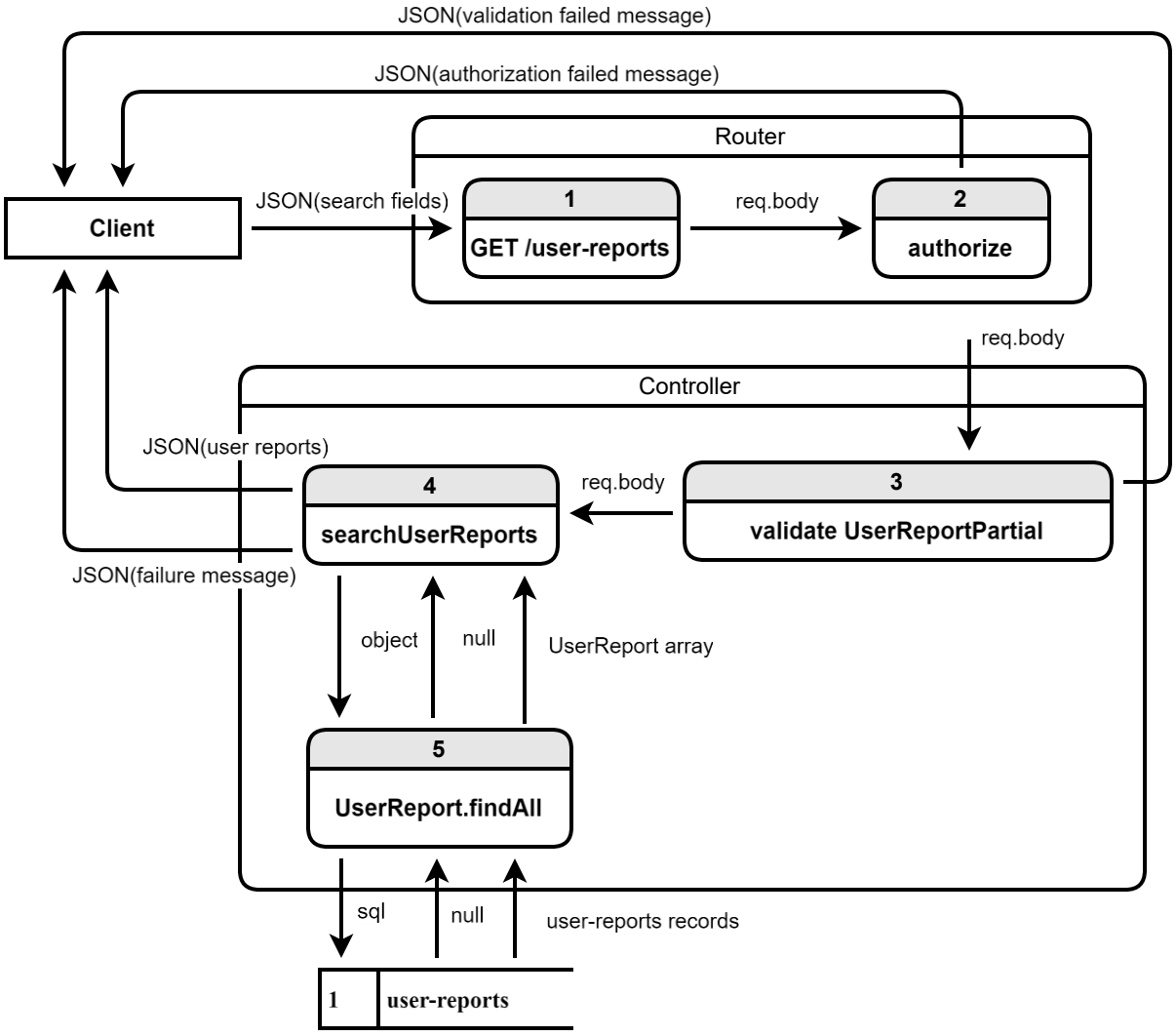
	1-2, 2-3, 3-4 {id} req.body The id is passed as a parameter in the url while req.body contains a JSON object with the updates to be applied to the event that corresponds to the id.
	4-5 object Java object with the updated fields
	5-group-events SQL query
	group-events-5 SQL query results
	5-4 group-event model A sequelize model of the newly updated group event
	4-client A JSON string on the newly updated group event
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.9.4 User Report Delete Endpoint



Name	1.1.3.1.9.4 User Report Delete Endpoint
Purpose	Show the flow of data when a client deletes a user report
Description	When a client sends a DELETE request to /user-reports, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data will be removed from the database and a JSON response confirmation will be sent to the client.
Requirements	LDR.7, LDV.7
Elements	**1.1.3.1.8.1 Router /user-reports** The route entry point to handle user reports
	1.1.3.2 Authorize Checks if the user is authorized to use this resource
	1.1.2.2.9.4 Controller User Report deleteUserReport The controller function responsible for deleting user reports.
	1.1.2.3.9 Model UserReport Sequelize model of user reports
	1.1.1.7 user-reports table Database table for user reports
	Client-1, 1-2, 2-3 ,3-4 {id} The id of the user report to be deleted
	4-user-reports SQL query to delete the user report
	user-reports-4, 4-3 rows deleted A number indicating the rows deleted
	3-Client JSON A message indicating the success or failure of the operation
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

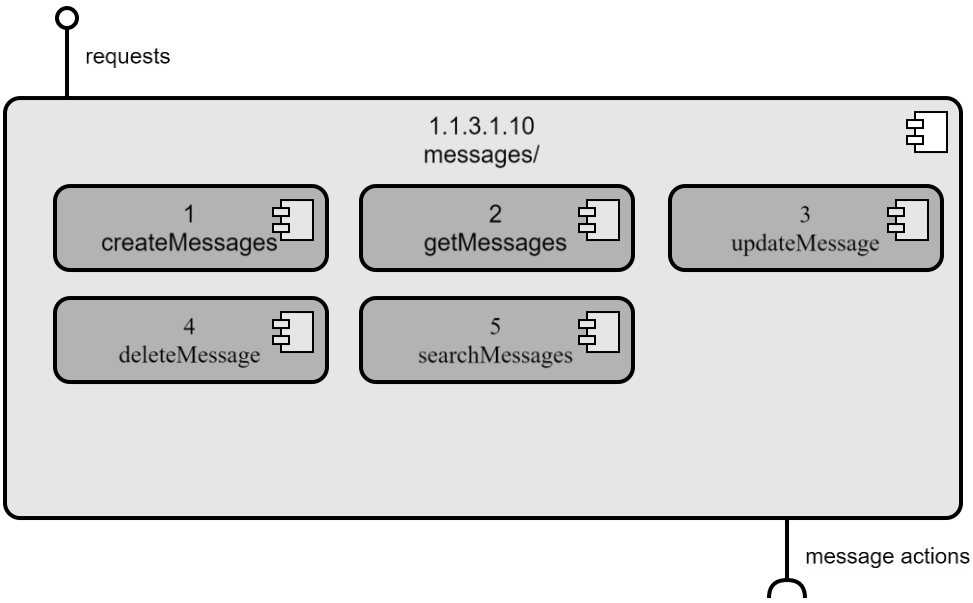
View 1.1.3.1.9.5 User Report Get Search Endpoint



Name	1.1.3.1.9.5 User Report Get Search Endpoint
Purpose	Show the flow of data when a client searches for a user report
Description	When a client sends a GET request to /user-report, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, that data is used as a field match to retrieve user report data. On a successful query, a JSON response with the data will be returned to the client. On a failed query, the failure message is sent.
Requirements	LDR.7, LDV.7
Elements	**1.1.3.1.9.5 Router /User Reports** The route entry point to handle user reports
	1.1.3.2 Authorize Checks user authorization
	1.1.2.1.15 Validation validateUserReportPartial Middleware that validates the fields of a user report. Does not require all fields be present
	1.1.2.2.9.5 Controller User Report searchUserReport The controller function responsible for searching for user reports based on a search fields

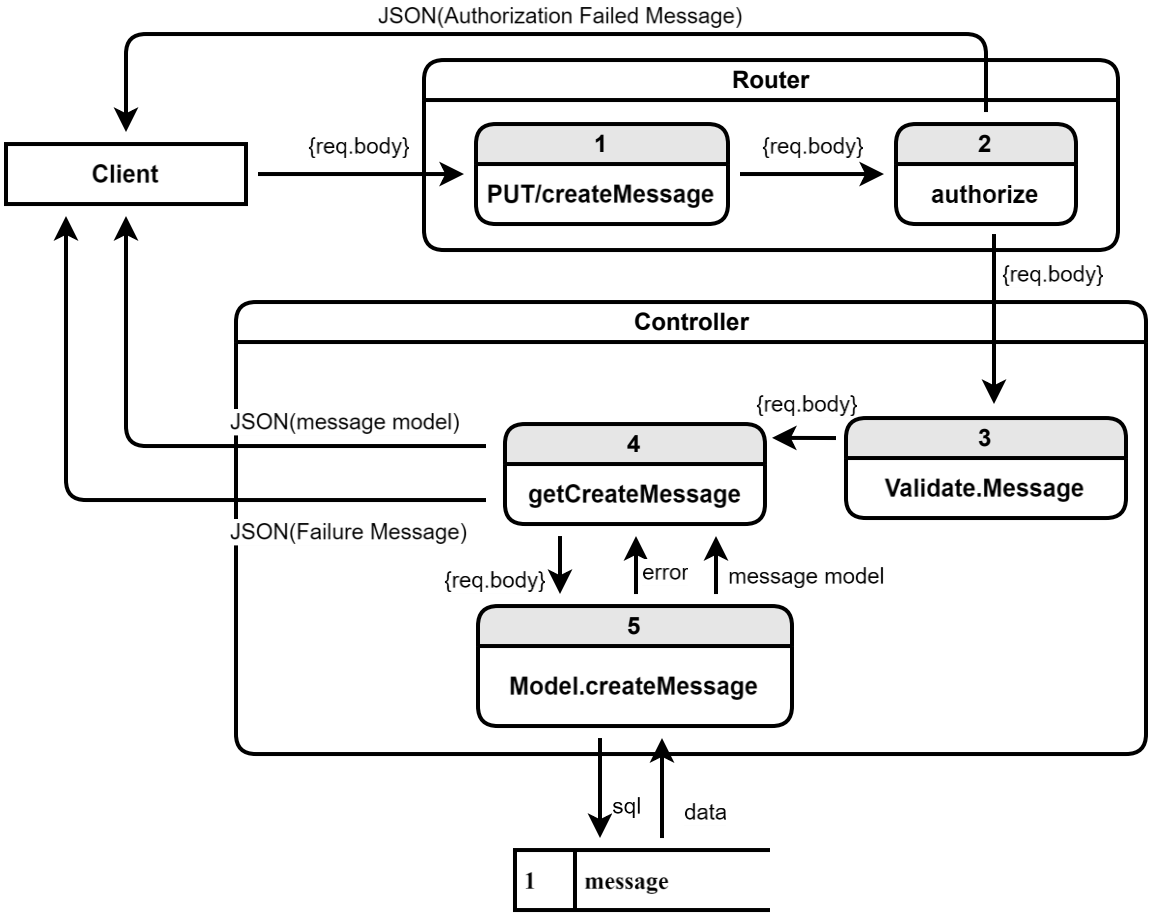
	1.1.2.3.9 Model User Report
	1.1.1.7 user-reports table Sequelize model of user reports
	Client-1 JSON JSON string in the HTTP request body field.
	1-2, 2-3, 3-4 req.body The standard Express request object. It may contain a body object which here will be a JavaScript object representation of a JSON string passed by the client. It can also contain a param object which contains parameters passed as part of the URL, most often an id field.
	4-5 object Java object containing the fields to be searched
	5-user-reports sql SQL query to find matching user report records
	user-reports-5 user-reports records SQL result set containing all the matching user reports
	5-4 UserReport array An array of UserReport models with the matching user reports
	4-Client JSON JSON string containing the list of user reports
Referenced by	1.1.3.1
Viewpoint	Data Flow Diagram

View 1.1.3.1.10: Sub Route Message



Name	1.1.3.1.10 Messages Endpoints
Purpose	To show the messages routes and endpoints through the UMessages application.
Description	When a client sends a Messages request, this view lists the different routes that the request will take based on the URL and POST/GET/PUT reqests.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**REQUEST** a request from the front end.
	1.1.3.1.6 Messages, the Messages router.
	1 Route to create a Message.
	2 Route to get Messages.
	3 Route to update a Message.
	4 Route to delete a Message.
	5 Route to search for a Message.
	Messages Actions, the various functions and actions for each route.
Referenced by	1.1.3.1
Viewpoint	Component Diagram

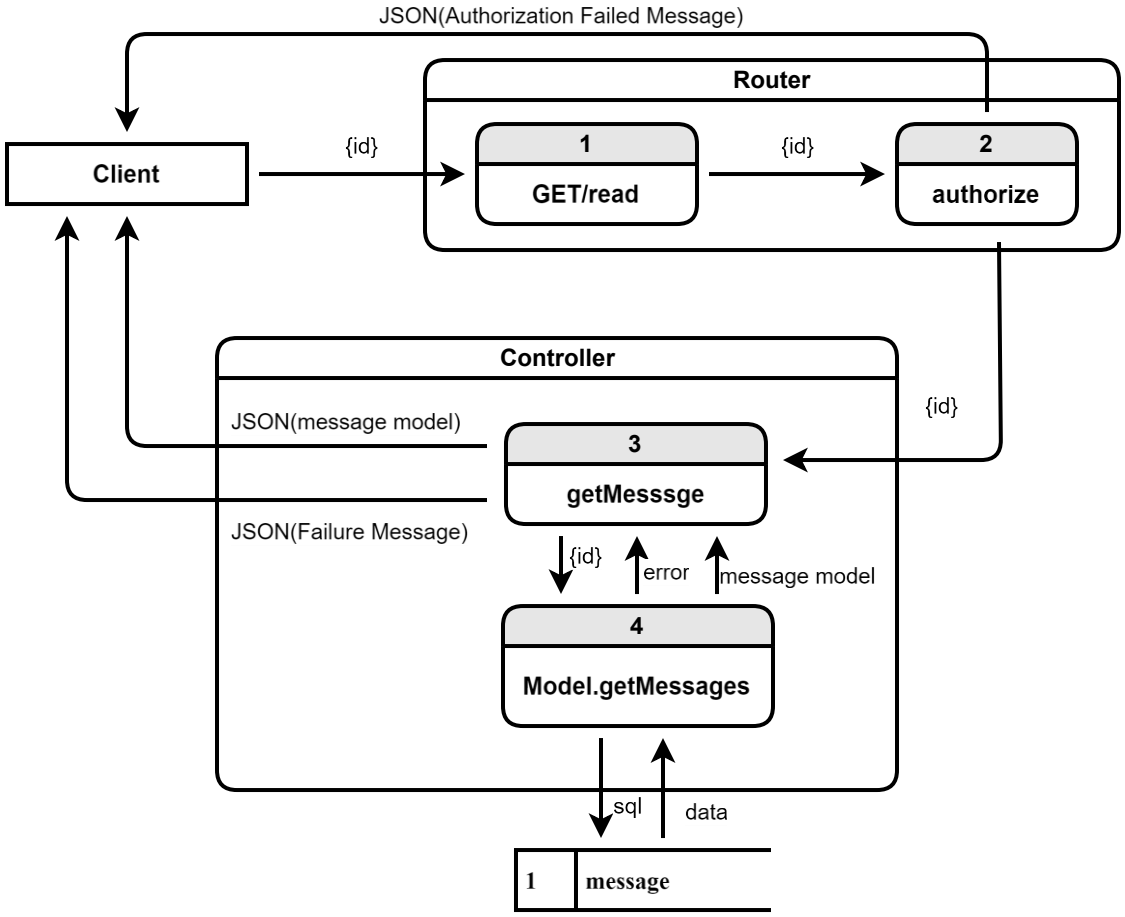
View 1.1.3.1.10.1: Commercial Messages Accounts POST



Name	1.1.3.1.10.1 Message POST(create)
Purpose	To show how the data flows through the backend when the user requests to make/send a message on Uevents.
Description	When a client sends a POST request to /message, the id and contents of the message sent pass through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with group event data will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Message
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller message
	1.1.2.3.X Message Model
	Client-1, 1-1, 2-1, 3-1 The id and contents of the message.
	2-2 JSON return to client if authorization fails.
	3-2 returns the message status as a JSON object.
	3-3 returns a failure message as a JSON object.

	4-1 SQL request to insert the message to the database
	4-2 returns an error from the model
	4-3 message model returns all the data to be added to the JSON object.
	1.1.1.X message table
	1-1 data return from the SQL query
Referenced by	1.1.10.1
Viewpoint	Data Flow Diagram

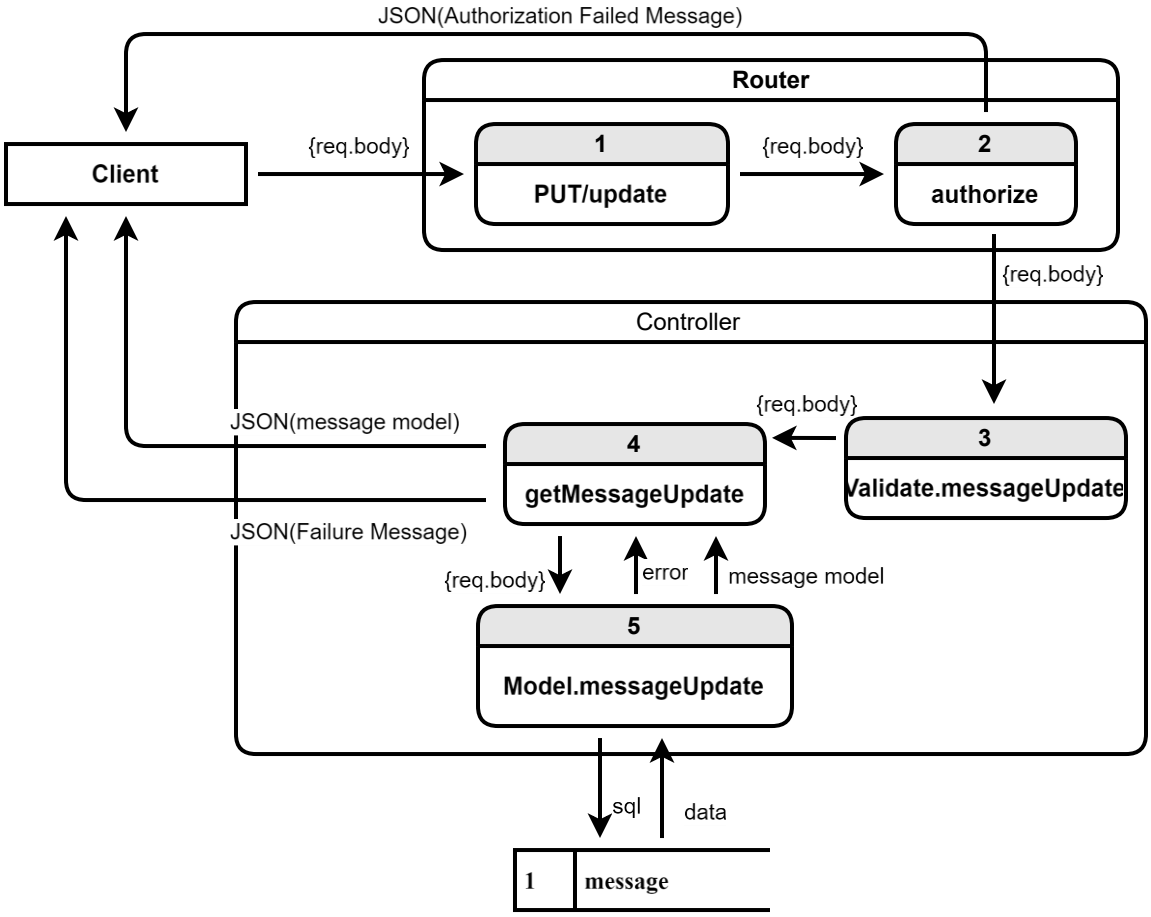
View 1.1.3.1.10.2: Messages GET



Name	1.1.3.1.10.2 Message Get(read)
Purpose	To show how the data flows through the backend when the user requests an message on Uevents.
Description	When a client sends a GET request to /message, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data will be queried from the database and a JSON response with message data will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Message
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller getMessage
	1.1.2.3.X Message Model
	Client-1, 1-1, 2-1, 3-1 The id of the message being requested.
	2-2 JSON return to client if authorization fails.
	3-2 returns the message data as a JSON object.
	3-3 returns a failure message as a JSON object.
	4-1 SQL request to get the message data from the database

	4-2 returns an error from the model
	4-3 event model returns all the data as the JSON object.
	1.1.1.X message table
	1-1 data return from the SQL query
Referenced by	1.1.10.1
Viewpoint	Data Flow Diagram

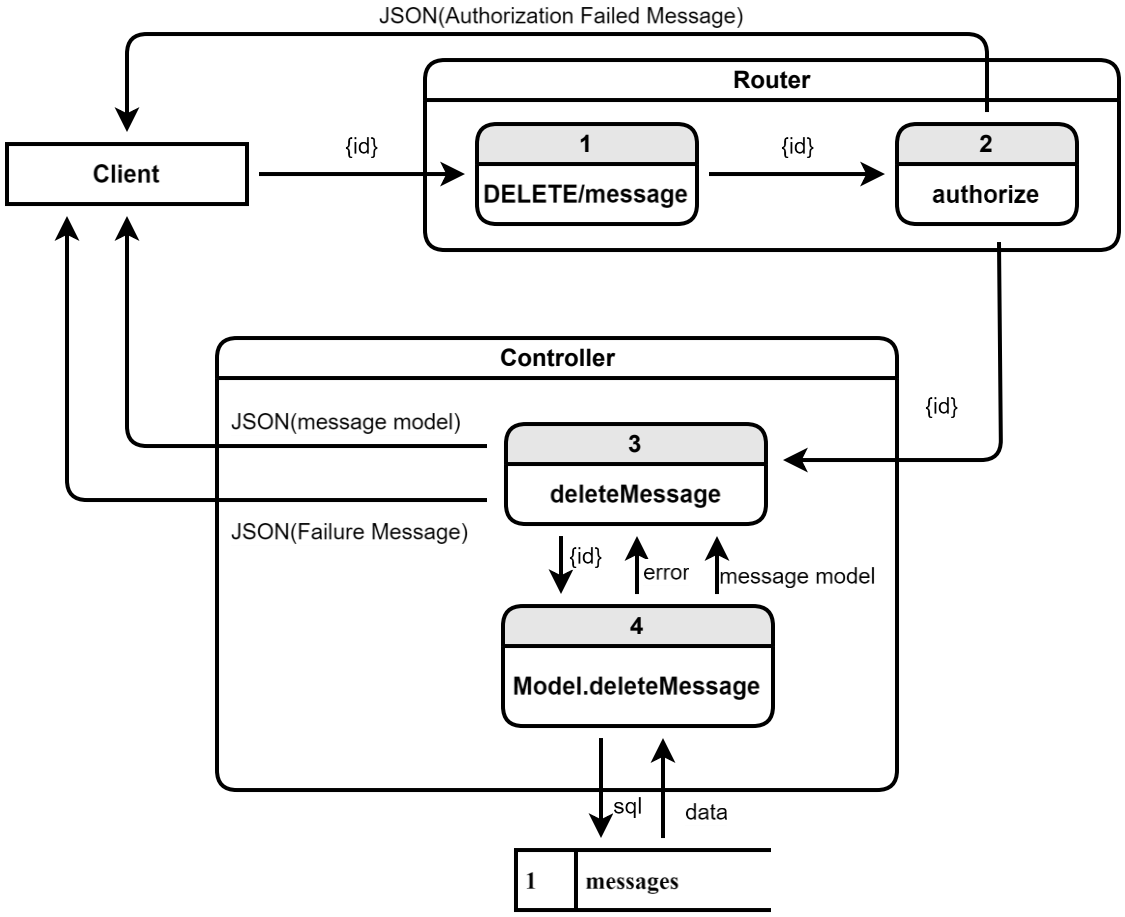
View 1.1.3.1.10.3: Messages PUT



Name	1.1.3.1.10.3 Messages PUT(update)
Purpose	To show how the data flows through the backend when the user requests to make an update/edit to a message on Uevents.
Description	When a client sends a PUT request to /events, the id of the message sent passes through the router and then through the rest of the system. If everything is done correctly, the data in the database will be updated and a JSON response summary / update will be returned to the client.
Requirements	FS.1, FS.4, FS.5, FS.6
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Message
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller updateMessages
	1.1.2.3.X Message Model
	Client-1, 1-1, 2-1, 3-1 The id of the message being updated.
	2-2 JSON return to client if authorization fails.
	3-2 returns the event data as a JSON object.
	3-3 returns a failure message as a JSON object.
	4-1 SQL request to update the message data in the database

	4-2 returns an error from the model
	4-3 event model returns a summary of the message update as a JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.10.1
Viewpoint	Data Flow Diagram

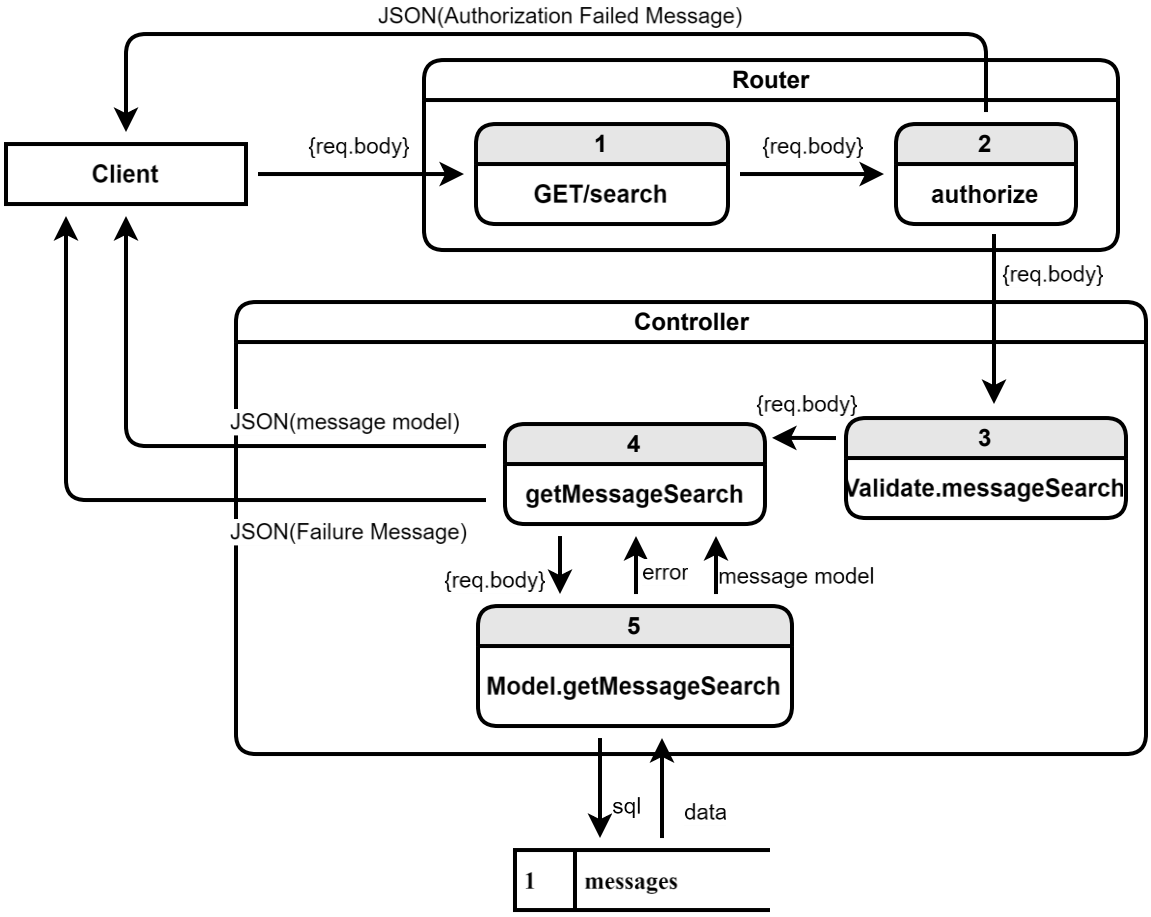
View 1.1.3.1.10.4: Messages DELETE



Name	1.1.3.1.10.4 Messages DELETE (delete)
Purpose	To show how the data flows through the backend when the user requests to delete a message on Uevents.
Description	When a client sends a DELETE request to /messages, the request sent passes through the router and then through the rest of the system. If everything is done correctly, the message will be deleted from the database and a JSON response will be returned to the client.
Requirements	PO.1.2.5, PO.1.7.23, PO .2 .1.12, FS.10
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Messages
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller deleteMessage
	1.1.2.3.X Model message
	Client-1, 1-1, 2-1, 3-1, 4-1 The id of the message to be deleted in a JSON object.
	2-2 JSON return to client if authorization fails.
	4-2 returns the results of the delete request as a JSON object.
	4-3 returns a failure message as a JSON object.

	5-1 SQL request summery of the result from the database
	5-2 returns an error from the model
	5-3 event model returns summary of the result as a JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.10.1
Viewpoint	Data Flow Diagram

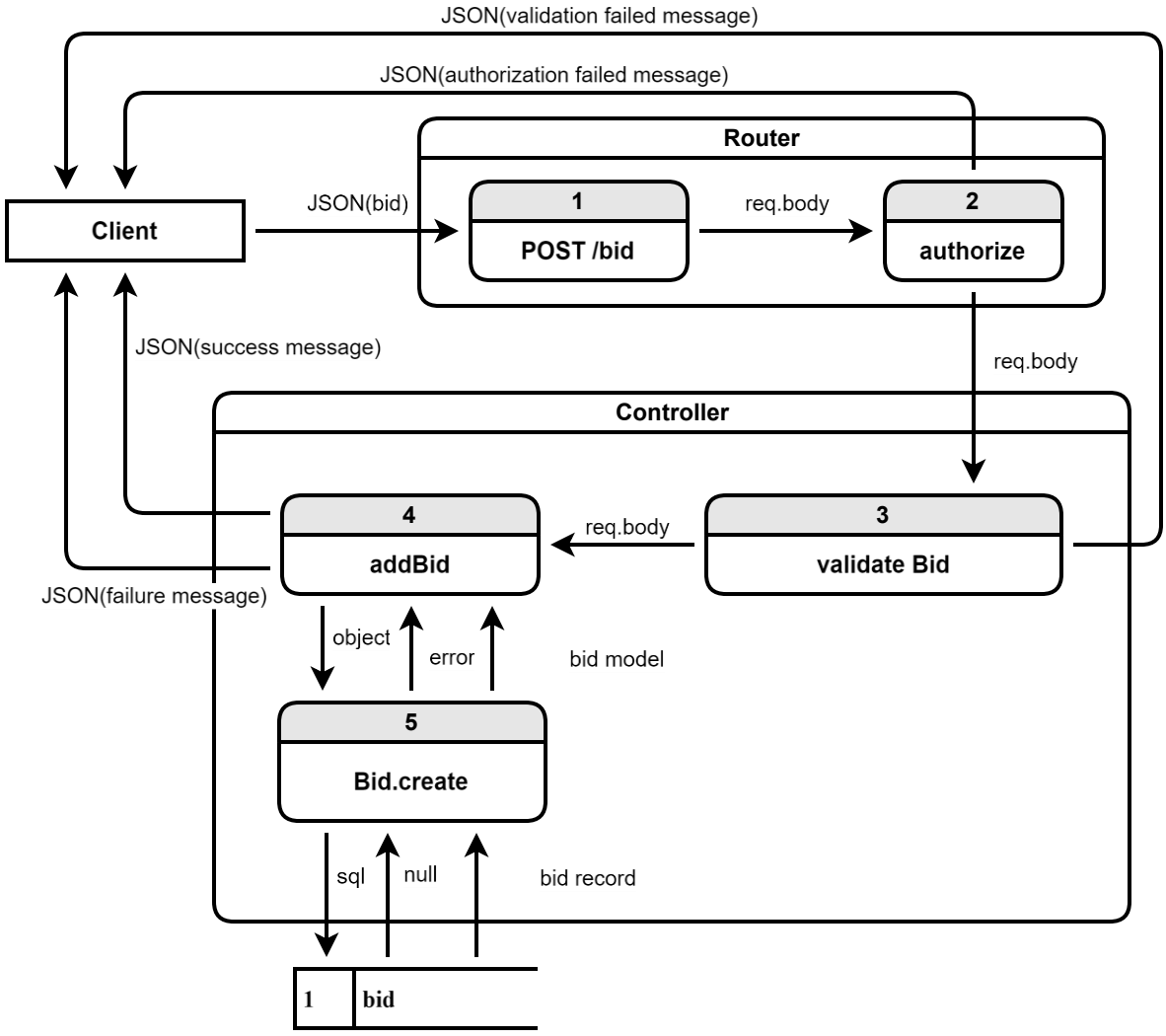
View 1.1.3.1.10.5: Message SEARCH



Name	1.1.3.1.10.5 Messages GET(Search)
Purpose	To show how the data flows through the backend when the user requests an messages search on Uevents.
Description	When a client sends a GET search request to /messages, the request sent passes through the router and then through the rest of the system. If everything is done correctly, the request will be used to query the database and a JSON response will be returned to the client.
Requirements	PO.1.2.5, PO.1.7.23, PO .2 .1.12, FS.10
Elements	**Client** a request from the front end
	1.1.3.1.8.1 Router / Messages
	1.1.3.2 Authorize
	1.1.2.2.X.X Controller searchMessage
	1.1.2.3.X Model message
	Client-1, 1-1, 2-1, 3-1, 4-1 The search request in a JSON object.
	2-2 JSON return to client if authorization fails.
	4-2 returns the message search results as a JSON object.
	4-3 returns a failure message as a JSON object.
	5-1 SQL request to get the message data from the database

	5-2 returns an error from the model
	5-3 event model returns all the matching data as a JSON object.
	1.1.1.X events table
	1-1 data return from the SQL query
Referenced by	1.1.10.1
Viewpoint	Data Flow Diagram

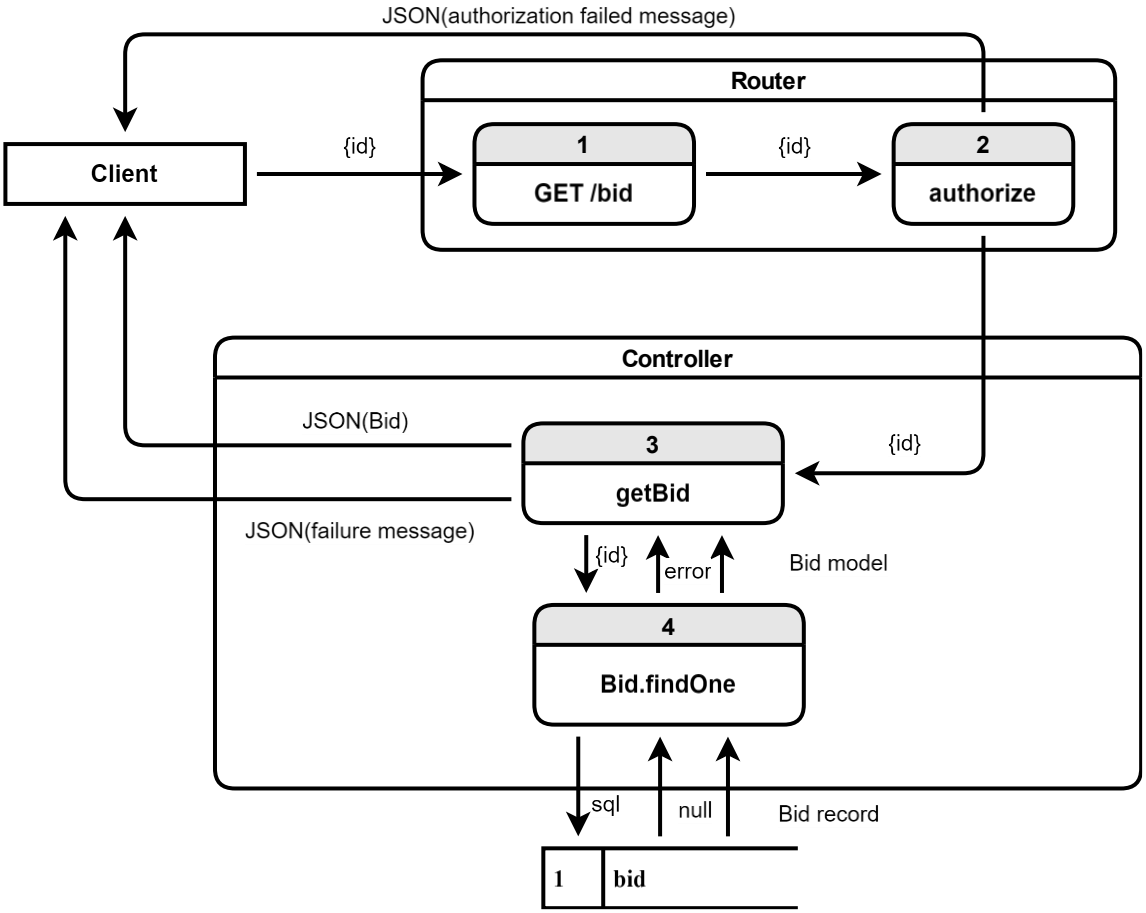
View 1.1.3.1.11.1 Bid Post Endpoint



Name	1.1.3.1.11.1 Bid Post Endpoint
Purpose	Show the flow of data when a client creates a bid
Description	When a client sends a POST request to /bid, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response will be returned to the client.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**1.1.3.1.11 Router /bid:** The route entry point to handle bids
	1.1.3.2 Authorize Checks if the user is authorized to use this resource
	1.1.2.1.11 Validation validateBid Middleware that validates the fields of a bid
	1.1.2.2.9.1 Controller Bid addBid The controller function that manages inserting the bids in to the database and creating a http response to the client.
	1.1.2.3.12 Model Bid Sequelize model of bids
	1.1.1.12 bid table Database table for bids

	Client-1 JSON JSON string in the HTTP request body field.
	1-2, 2-3, 3-4 Req The standard Express request object. It may contain a body object which here will be a JavaScript object representation of a JSON string passed by the client. It can also contain a param object which contains parameters passed as part of the URL, most often an id field.
	4-5 object JavaScript object containing the bid data
	5-bid sql Sequelize generated SQL query to create the bid record
	bid-5 record The result of the SQL query, or null if failed.
	1.1.2.3.12 5-4 bid model An instance of the bid model containing the newly created data
	4-client success/fail message A JSON string with a message field stating the API call was successful or not
Referenced by	1.1.3.1.11
Viewpoint	Data Flow Diagram

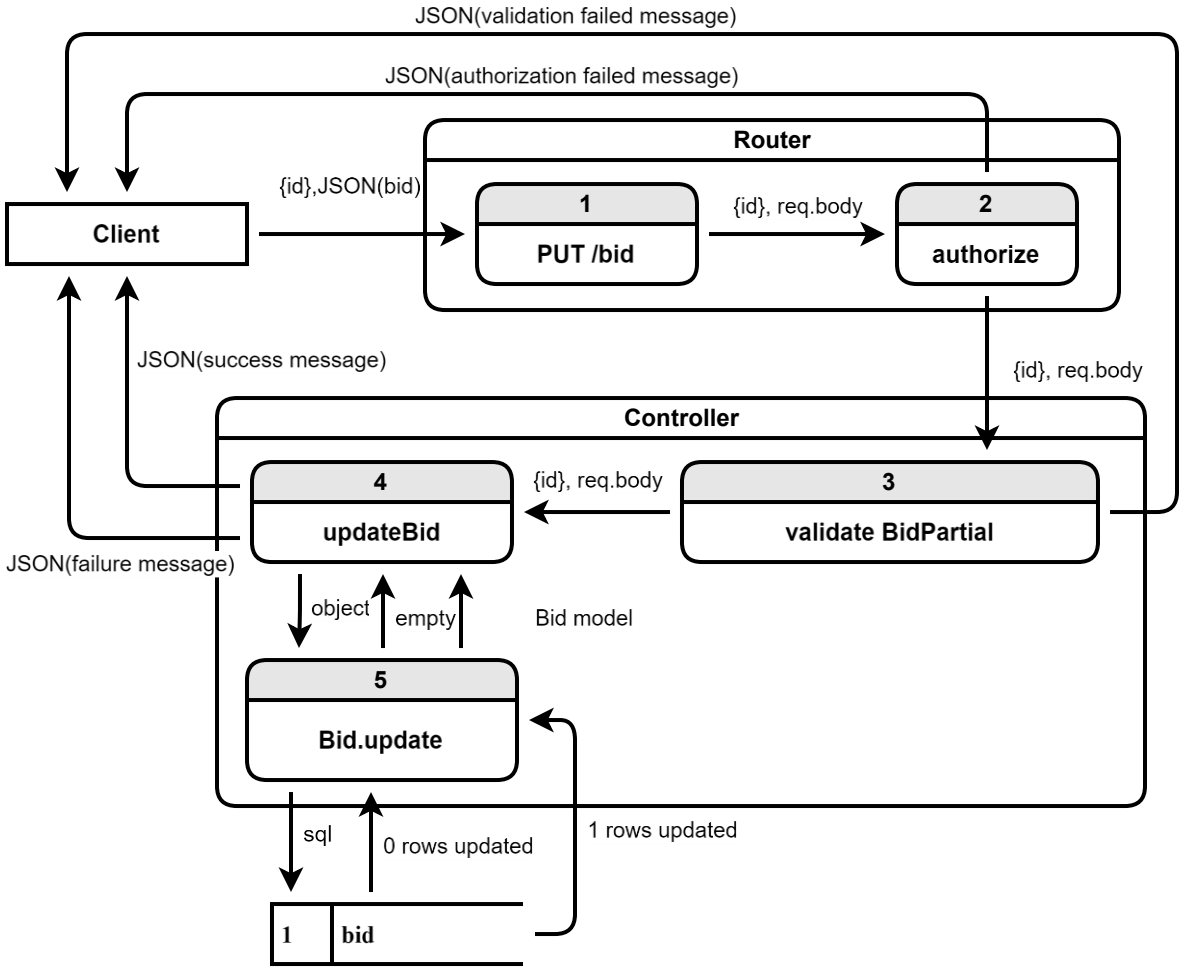
View 1.1.3.1.11.2 Bid Get Endpoint



Name	1.1.3.1.11.2 Bid Get Endpoint
Purpose	Show the flow of data when a client retrieves a bid
Description	When a client sends a GET request to /bid, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the new data will be stored in the database and a JSON response with bid data will be returned to the client.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**1.1.3.1.11.1 Router /bids** The route entry point to handle bids
	1.1.3.2 Authorize Module to authorize users to access this resource
	1.1.2.2.11.1 Controller Bid getBid
	1.1.2.3.12 Model Bid Sequelize model of bids
	1.1.1.12 bid table Database table for bids
	Client-1 id ID number parameter for the bid
	1-2, 2-3 id ID number passed through the req.param.id field
	3-4 id ID passed as an integer
	4-bid sql SQL query to retrieve the bid
	bid-4 bid record SQL query result containing the bid

	1.1.2.3.12 4-3 Bid model An instance the the Bid model containing the desired bid or error.
	3-Client Bid A JSON object string containing the requested bid or a failure message
Referenced by	1.1.3.1.11
Viewpoint	Data Flow Diagram

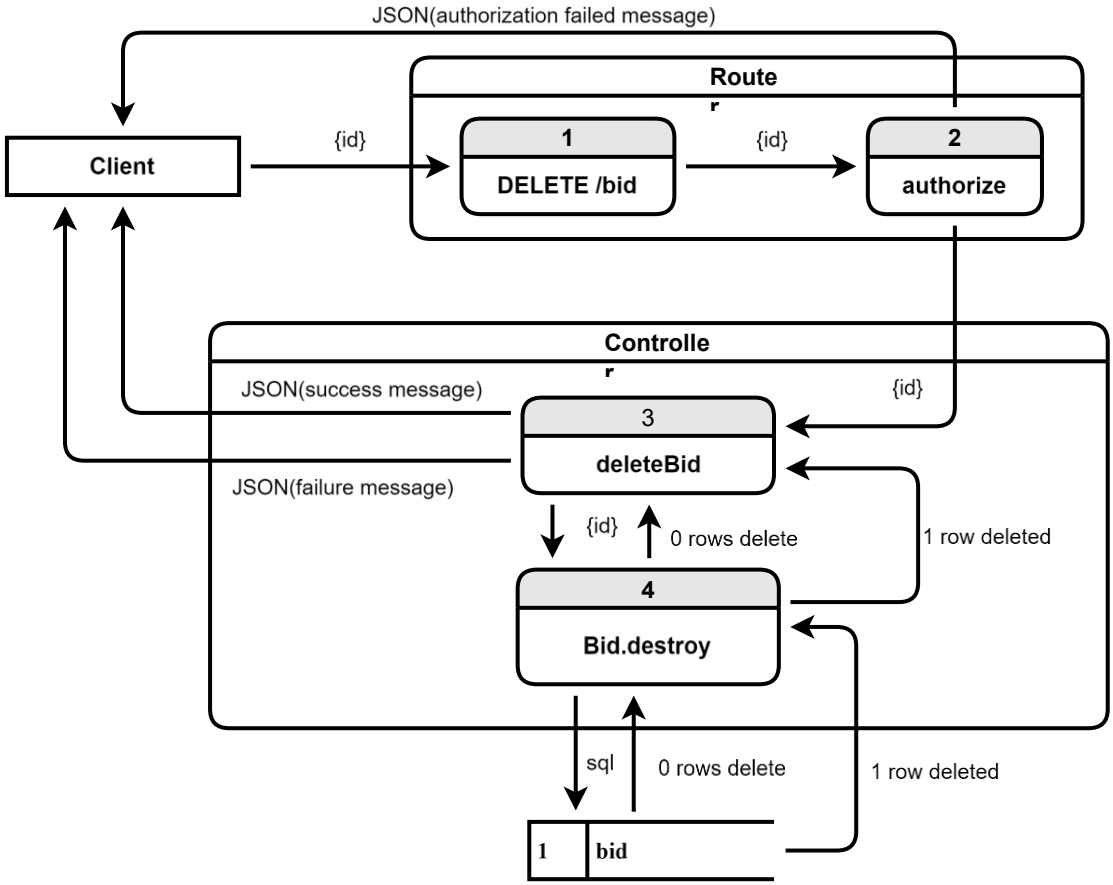
View 1.1.3.1.11.3 Bid Put Endpoint



Name	1.1.3.1.11.3 Bid Put Endpoint
Purpose	Show the flow of data when a client updates a bid
Description	When a client sends a PUT request to /bid, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, the new bid data will be updated in the database and a JSON response will be returned to the client.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**1.1.3.1.11.1 Router /bids** The route entry point to handle bids
	1.1.3.2 Authorize Module to authorize users to access this resource
	1.1.2.2.11.1 Controller Bid updateBid Controller method to update a bid
	1.1.2.3.12 Model Bid Sequelize model of bids
	1.1.1.12 bids table Database table for bids
	Client-1 {id} JSON id is passed as a parameter with the request URL while that updated fields are in the body as a JSON document
	1-2, 2-3, 3-4 {id} req.body The id is passed as a parameter in the url while req.body contains a JSON object with the updates to be applied to the bid that corresponds to the id.

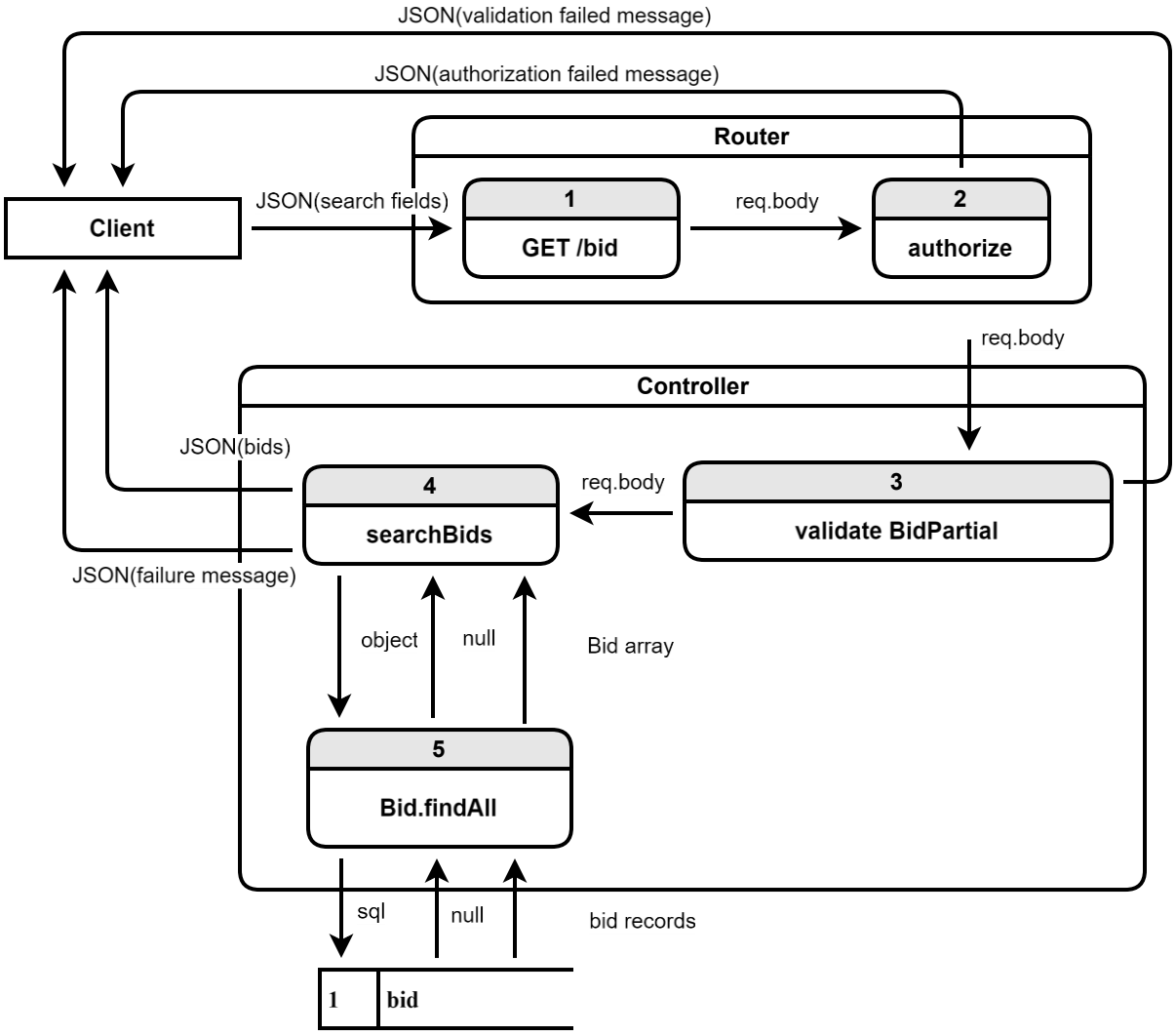
	4-5 object Java object with the updated fields
	5-bid SQL query
	bid-5 SQL query results
	1..1.2.3.12 5-4 bid model A sequelize model of the newly updated bid
	4-client A JSON string on the newly updated bid
Referenced by	1.1.3.1.11
Viewpoint	Data Flow Diagram

View 1.1.3.1.11.4 Bid Delete Endpoint



Name	1.1.3.1.11.4 Bid Delete Endpoint
Purpose	Show the flow of data when a client deletes a bid
Description	When a client sends a DELETE request to /bids, the id sent passes through the router and then through the rest of the system. If everything is done correctly, the data will be removed from the database and a JSON response confirmation will be sent to the client.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**1.1.3.1.11.1 Router /bids** The route entry point to handle bids
	1.1.3.2 Authorize Checks if the user is authorized to use this resource
	1.1.2.2.11.4 Controller Bid deleteBid The controller function responsible for deleting bids.
	1.1.2.3.12 Model Bid Sequelize model of bids
	1.1.1.12 bids table Database table for bids
	Client-1, 1-2, 2-3 ,3-4 {id} The id of the bid to be deleted
	4-bids SQL query to delete the bid
	bids-4, 4-3 rows deleted A number indicating the rows deleted
	3-Client JSON A message indicating the success or failure of the operation
Referenced by	1.1.3.1.11
Viewpoint	Data Flow Diagram

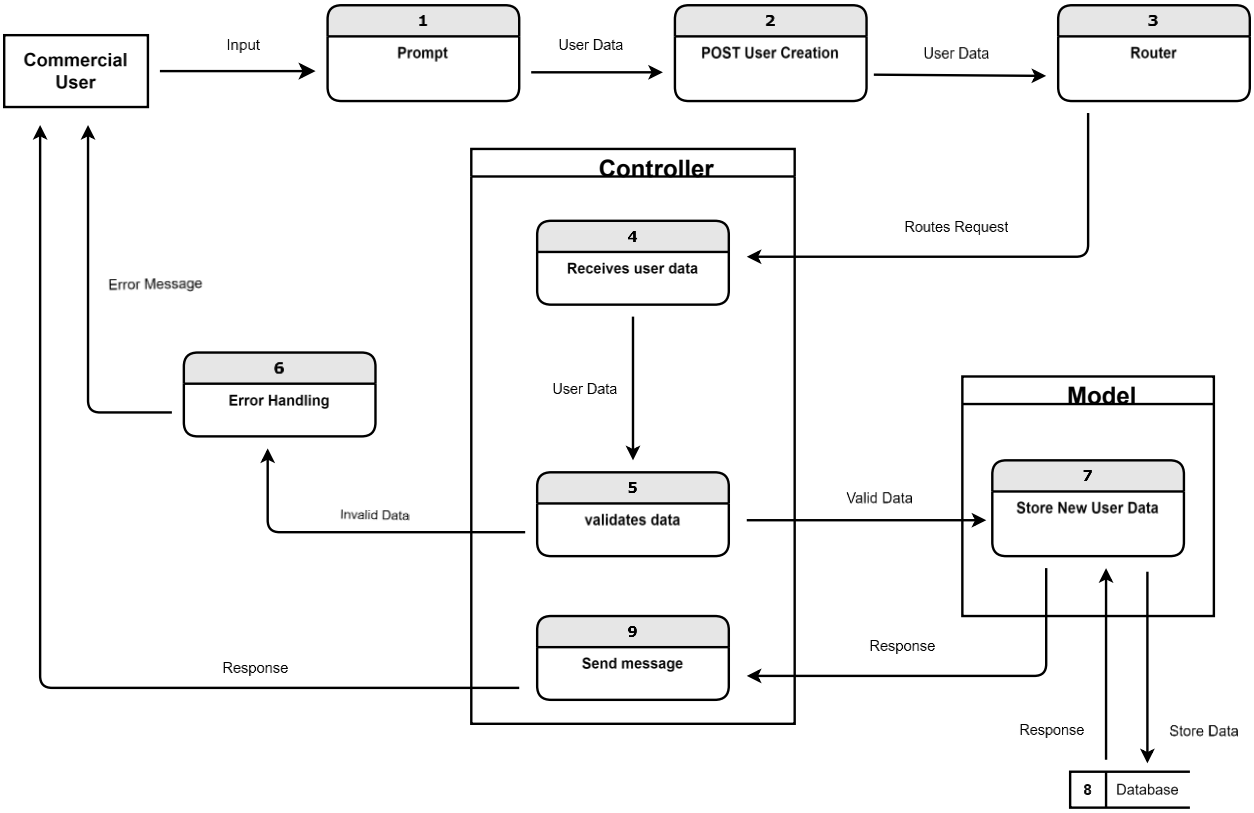
View 1.1.3.1.11.5 Bid Get Search Endpoint



Name	1.1.3.1.11.5 Bid Get Search Endpoint
Purpose	Show the flow of data when a client searches for a bid
Description	When a client sends a GET request to /bid, the JSON data sent passes through the router and then through the rest of the system. If everything is done correctly, that data is used as a field match to retrieve bid data. On a successful query, a JSON response with the data will be returned to the client. On a failed query, the failure message is sent.
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	**1.1.3.1.11.5 Router /Bids** The route entry point to handle bids
	1.1.3.2 Authorize Checks user authorization
	1.1.2.1.17 Validation validateBidPartial Middleware that validates the fields of a bid. Does not require all fields be present
	1.1.2.2.11.5 Controller Bid searchBid The controller function responsible for searching for bids based on a search fields
	1.1.2.3.12 Model Bid

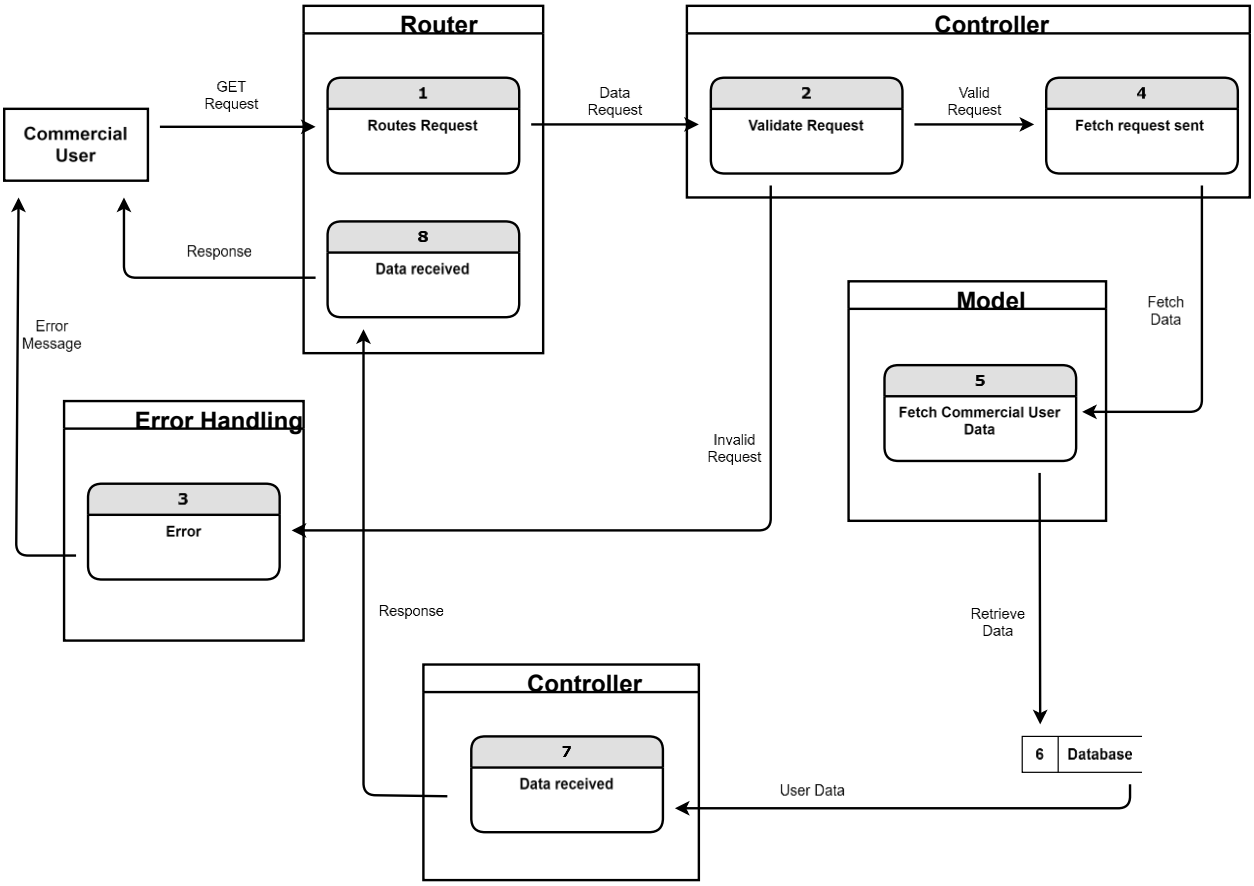
	1.1.1.12 bids table Sequelize model of bids
	Client-1 JSON JSON string in the HTTP request body field.
	1-2, 2-3, 3-4 req.body The standard Express request object. It may contain a body object which here will be a JavaScript object representation of a JSON string passed by the client. It can also contain a param object which contains parameters passed as part of the URL, most often an id field.
	4-5 object Java object containing the fields to be searched
	5-bids sql SQL query to find matching bid records
	bids-5 bids records SQL result set containing all the matching bids
	1.1.2.3.12 5-4 Bid array An array of Bid models with the matching bids
	4-Client JSON JSON string containing the list of bids
Referenced by	1.1.3.1.11
Viewpoint	Data Flow Diagram

View 1.1.3.1.12.1: Commercial User Accounts POST Data Flow Diagram



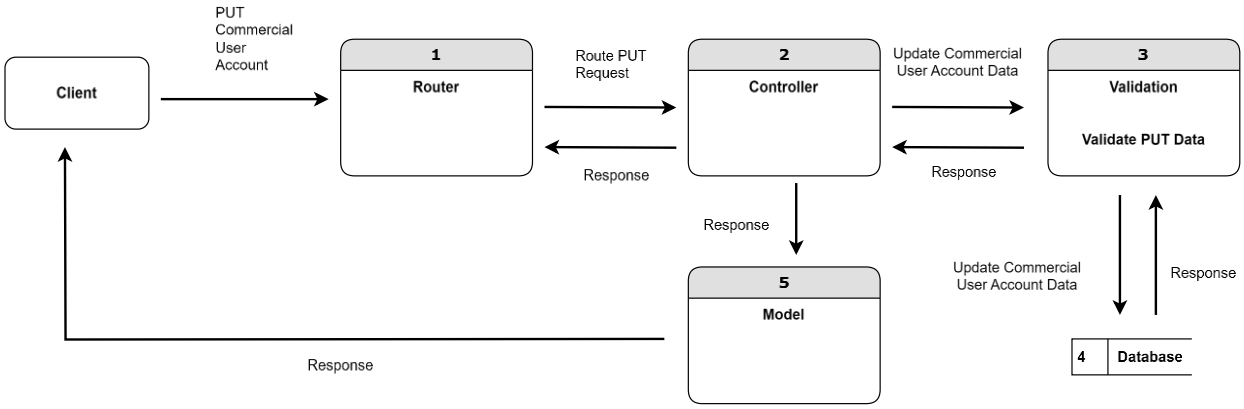
Name	1.1.3.1.12.1 Commercial User Accounts POST Data Flow Diagram
Purpose	Describe the POST process for a commercial user.
Description	This data flow describes the data flow when a commercial user registers through the POST method.
Requirements	PO.2.2.1, PO.3.2, PO.4.6
Elements	1. **Commercial User** is prompted to enter their data
	2. **POST** sends user data.
	3. **Router** receives user data then routes the request
	4. **Controller** receives user data
	5**. Controller** validates the data
	6. **Controller** sends error message to user if invalid data was sent
	7. **Model** receives valid data
	8. **Database** receives data to store from **Model** and sends response back
	9. **Controller** receives response from **Model** then send response to user
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.1.12.2: Commercial User Accounts GET Data Flow Diagram



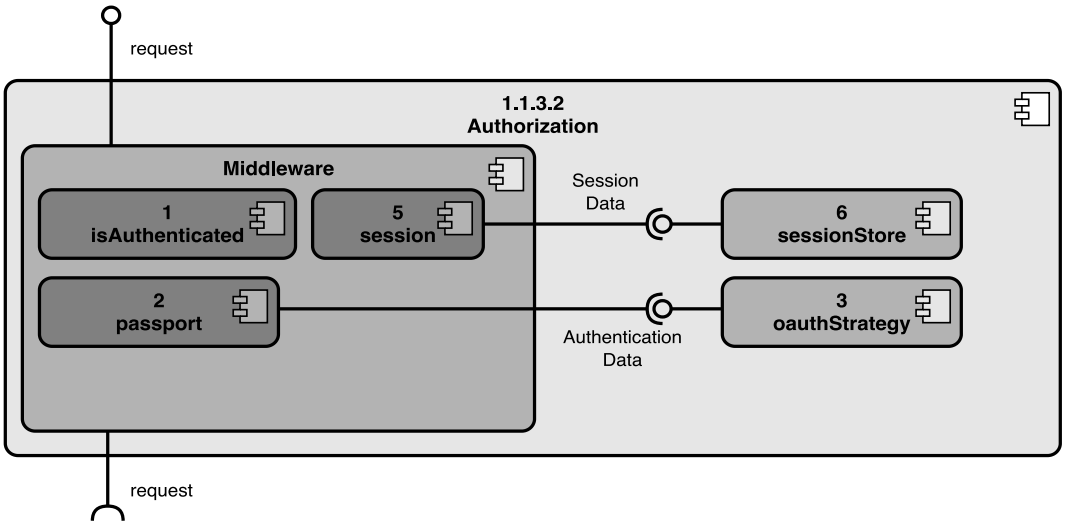
Name	1.1.3.1.12.2 Commercial User Accounts GET Data Flow Diagram
Purpose	Describe the GET process to retrieve data.
Description	This data flow describes the process to retrieve user data using the GET method.
Requirements	PO.2.2.1, PO.3.2, PO.4.6
Elements	1. **Commercial User** sends request to the Router which then routes the request.
	2. **Controller** receives data request then validates request.
	3. **Controller** send error message to use if request is invalid
	4. If request is valid then Fetch request is sent to **Model**
	5. **Model** retrieves data from database.
	6. Database sends user data to Controller
	7. **Controller** sends response to **Router**
	8. **Router** sends response to commercial user
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.1.12.3: Commercial User Accounts PUT Data Flow Diagram



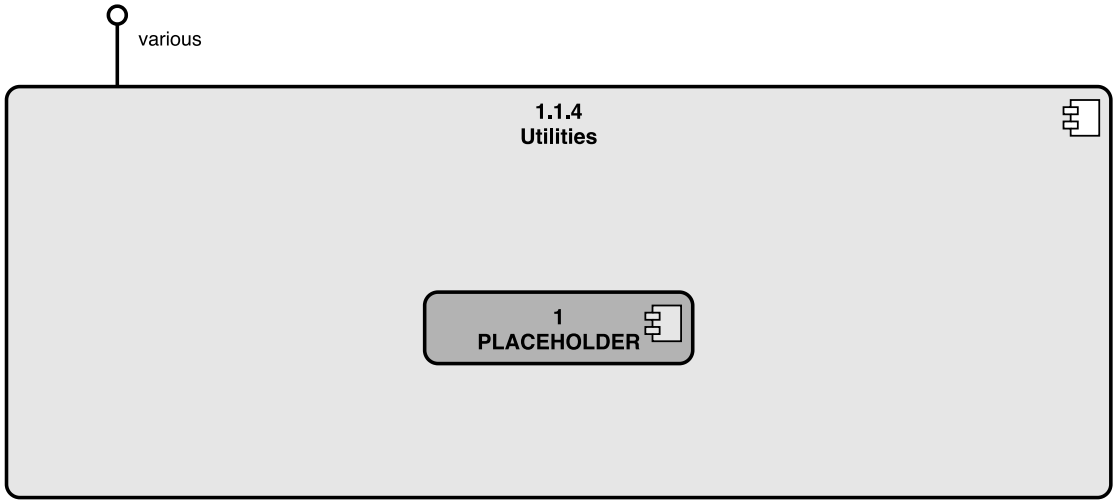
Name	1.1.3.1.12.3: Commercial User Accounts PUT Data Flow Diagram
Purpose	Provide the flow of data for updating commercial user accounts via a PUT request.
Description	System Operations: PUT Commercial User Account
Requirements	PO.1.7 System Operations: PUT Commercial User Account
Elements	**1.1.3.1 Router: Handles all API server routes.** **Router: Routes PUT request to the controller.** **1.1.2.2.1 Validation: Module that validates client data.** **Validation: Validates PUT data.** **1.1.2 Controller: Applies logic to the request and supplied data.** **Controller: Handles the PUT request and interacts with Validation and Model.** **1.1.2.2.3 Model: Model components that interface with the database.** **Model: Updates the commercial user account data in the database.** **Database: Stores and retrieves commercial user account data.** **Database: Updates and responds with the commercial user account data.**
Referenced by	1.1.3
Viewpoint	Data Flow Diagram

View 1.1.3.2 Authorization Module



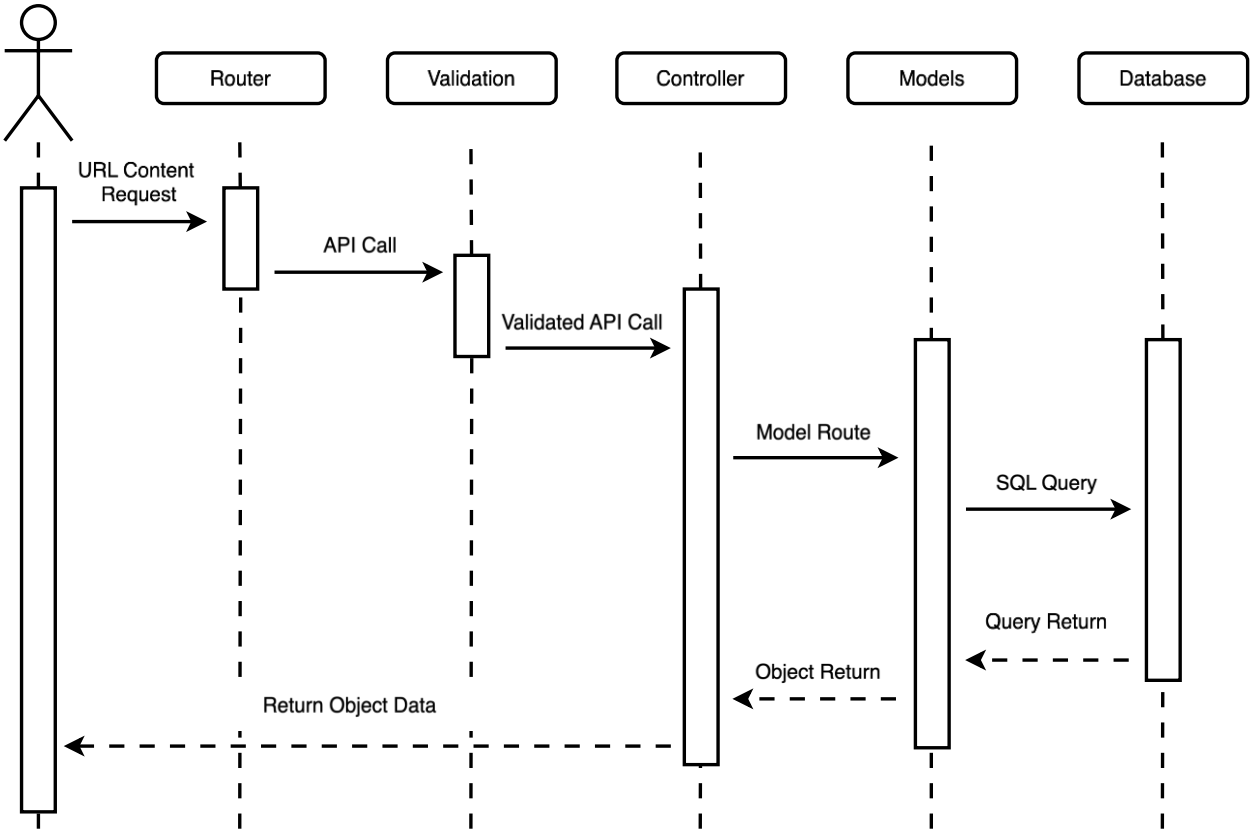
Name	1.1.3.2 Authorization Module
Purpose	To facilitate authorization and authentication concerns.
Description	This is a module that has several middleware functions to authenticate and process logged-in users. These rely on the oauthStratagy and sessionStore classes to function.
Requirements	FS.6, FS.14, FS.18, FS.25, FS.30, FS.38
Elements	**1.1.3.2.1 isAuthenticated** Middleware that checks if a user is authenticated
	1.1.3.2.2 Passport Implements middleware to facilitate authentication
	1.1.3.2.3 oauthStrategy A class that contains the oauth settings
	1.1.3.2.5 session Middleware that tracks a session and saves it to the database
	1.1.3.2.6 sessionStore A class to abstract session interactions with the database session table
Referenced by	1.1.3
Viewpoint	Component Diagram

View 1.1.4 Utilities Module



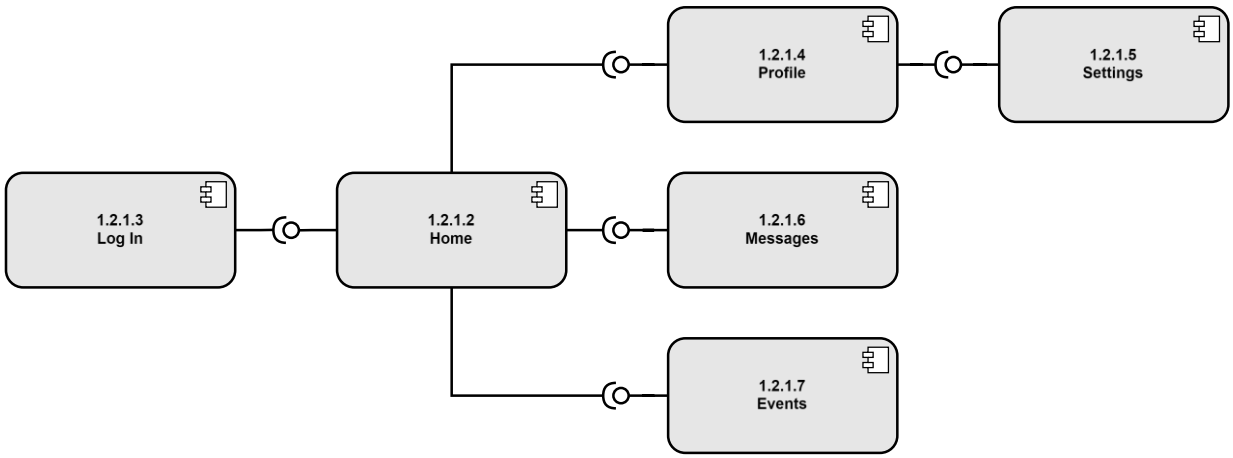
Name	1.1.4 Utilities Module
Purpose	A general-purpose module for utility functions.
Description	THIS IS CURRENTLY A PLACEHOLDER. When frequently used functionalities are encountered, they are placed here.
Requirements	To be determined
Elements	To be determined
Referenced by	1.1
Viewpoint	Component View

View 1.1.4.1: GET Request Sequence



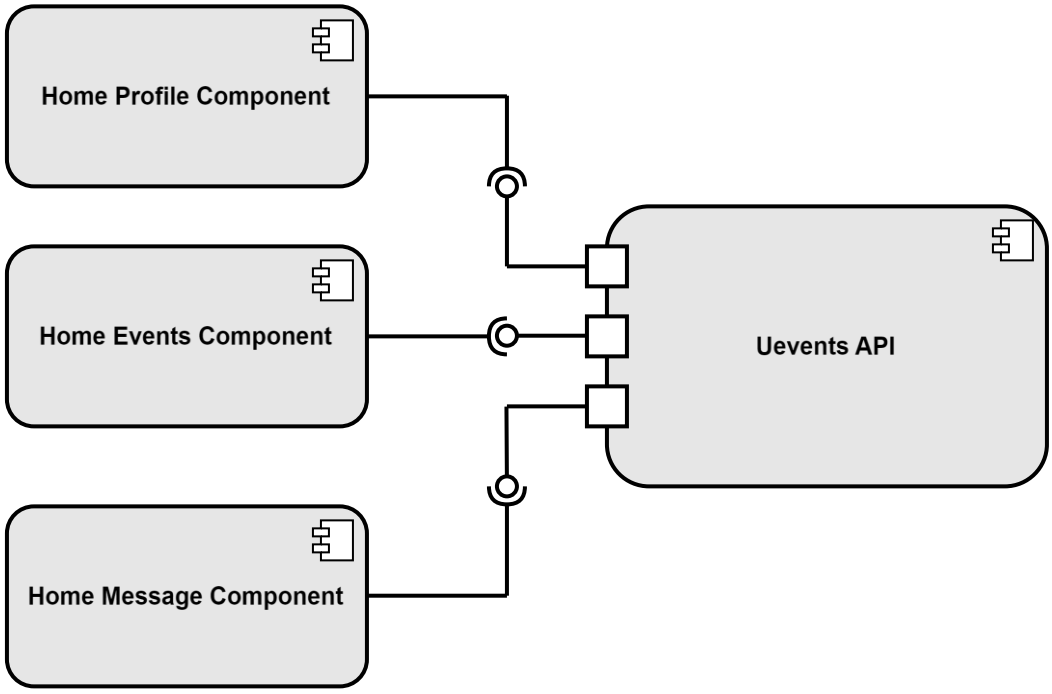
Name	**1.1.4.1 GET Request Sequence Diagram**
Purpose	Overview of the GET request
Description	This diagram demonstrates the sequence of a GET Request
Requirements	PO.1.1.4
Elements	**URL Content Request** **1.1.4.1 User** sends a **GET request** through the **URL** **1.1.4.2** The **Router** makes an **API call** to the **Controller** **1.1.4.3** The **Controller** routes the call to the **Model** **1.1.4.4 Model** sends a **SQL Query** to the **Database** **1.1.4.5 Database** sends a **Query** back to the **Model** **1.1.4.6 Model** sends a **JSON template** to the **Controller** **1.1.4.7** The **Controller** then sends the **JSON** Data back to the **User**
Referenced by	1.1.4
Viewpoint	Sequence Diagram

View 1.2.1.1: Mobile Components Master View



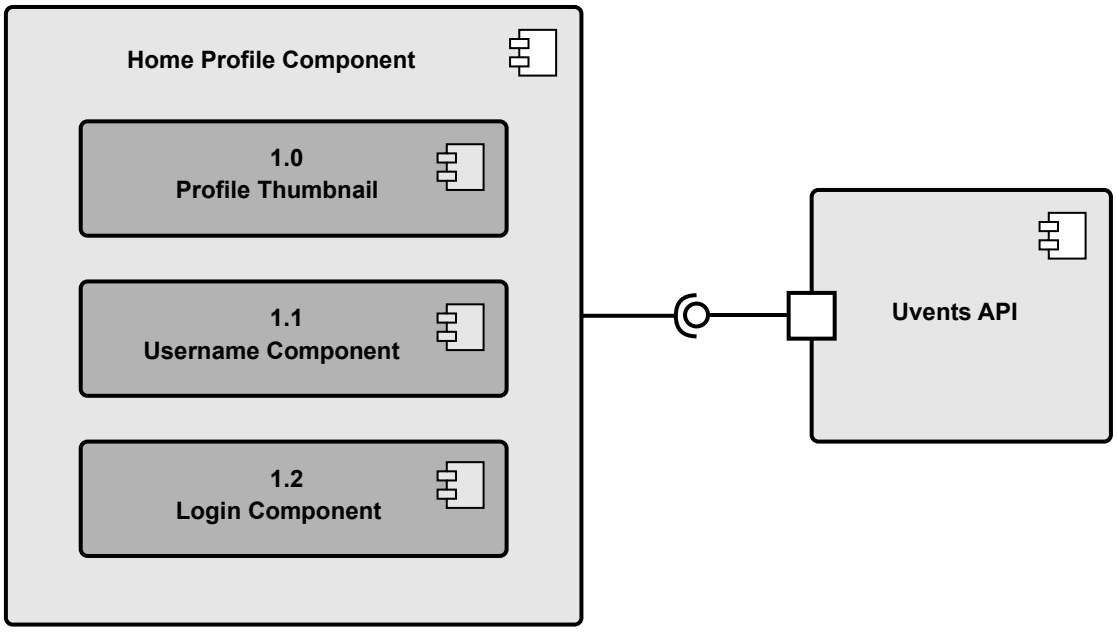
Name	1.2.1.1 Mobile Components Master View
Purpose:	Illustrates a high view of the structure of the components of the mobile application.
Description:	Shows the connection between all components within each mobile application of Uvents.
Requirements:	PO.1.2.1, PO.1.2.5, PO.1.2.12, PO.1.2.14, PO.1.2.15
Elements:	**1.2.1.3 Mobile Log In:** Provides functionality for users of different types to log in Uvents.
	1.2.1.2 Home: Provides a home view of Uvents.
	1.2.1.4 Profile: Provides profile functionality for users of Uvents.
	1.2.1.5 Settings: Provides application settings to the user.
	1.2.1.6 Messages: Provides messaging functionality for the user.
	1.2.1.7 Event Feed: Provides an event feed for the user.
Referenced By:	1
Viewpoint:	Component Diagram

View 1.2.1.2: Mobile Home View



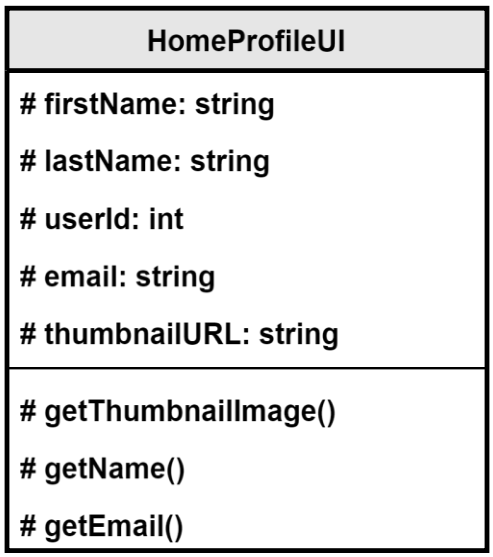
Name	1.2.1.2 Mobile Home View
Purpose:	The purpose of this document is to demonstrate the Mobile Home View component structure. The Mobile Home View provides users with a central location to access primary app features.
Description:	The Mobile Home View contains UI components that link to three primary app features. The features that can be accessed from the Mobile Home View are the Message View, Profile View, and Events View.
Requirements:	PO. 1.2.12
Elements:	**Home Message Component**: A UI component that contains a link to the app's Message View. Unread messages can change the appearance of the Home Message component to alert users of the new messages through the Uvents API.
	Home Profile Component: A UI component that contains a link to the app's Profile View. A thumbnail of the profile picture within the Home Profile Component helps the user identify what user is currently logged-in, if any, through the Uvents API.
	Home Events Component: A UI component that contains a link to the app's Events View. The Home Events Component contains a CTA directing users to join events. The appearance of the Home Events Component can change to alert users of new events, or changes to joined event details, through the Uvents API.
Referenced By:	1.2.1.1
Viewpoint:	Component Diagram

View 1.2.1.2.1 Mobile Home Profile UI



Name	1.2.1.2.1 Mobile Home Profile UI
Purpose:	Provides a link for the user to view their profile and verifies their login status
Description:	A UI component on the mobile version of the homepage containing sub-components for user navigation to the profile view and profile information. Retrieves session info and profile data using the Uvents API, which it passes to sub-components.
Requirements:	PO.1.2.12, PO.1.7.17, PO.1.7.21, PO.1.7.34, PO.3.4.2, PO.3.5.2, PO.3.6.2
Elements:	**1.0 Profile Thumbnail:** a UI component which links to the profile view. Link behavior and thumbnail appearance are dependent on login status.
	1.1 Username Component: a UI component which contains the user's first and last name if logged in, which also links to the profile view. Link behavior and username appearance are dependent on login status.
	1.2 Login Component: a UI component enabled only when no active session can be retrieved using the Uvents API, which links to the user login flow.
	Uvents API: an external service with endpoints for retrieving profile data and session info
Referenced By:	1.2.1.2
Viewpoint:	Component Diagram

View 1.2.1.2.1.1: Mobile Home Profile UI Class



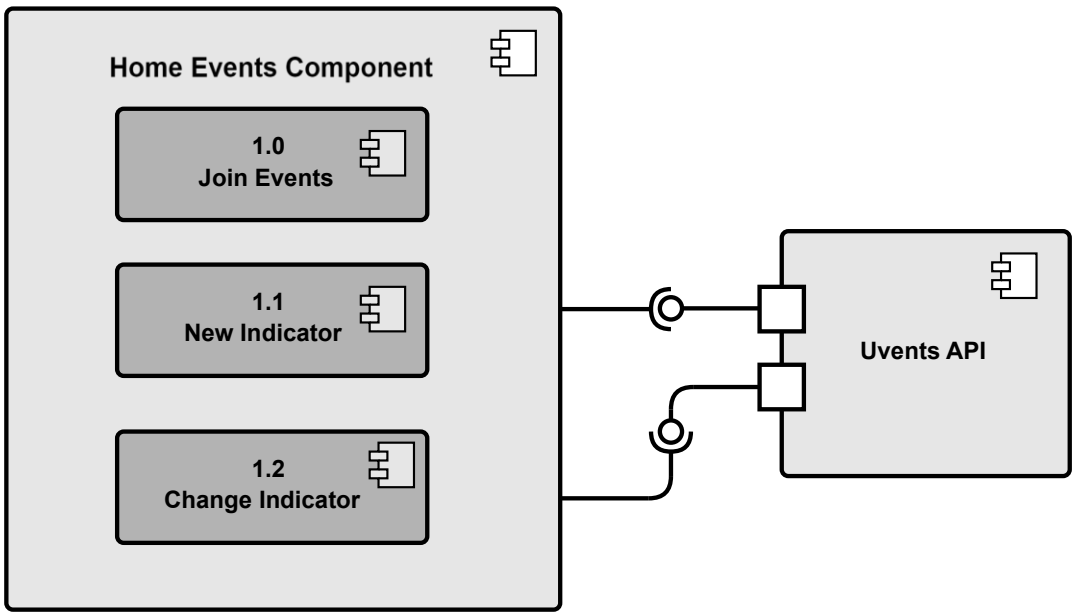
Name	**1.2.1.2.1.1 Mobile Home Profile UI Class Diagram**
Purpose:	Describe the classes involved in the mobile home profile UI
Description:	The Mobile Home Profile UI Component needs to retrieve a user's session information, such as their name, to display within the component.
Requirements:	PO.1.2.12, PO.1.7.17, PO.1.7.21, PO.1.7.34, PO.3.4.2, PO.3.5.2,PO.3.6.2
Elements:	**firstName:** identifies the user first name
	lastName: identifies the user last name
	userId: identifies the user
	email: identifies the user email
	thumbnailURL: identifies the user thumbnail URL
	getthumbnailImage: retrieves the thumbnail image from the DB and displays it on the Home Page.
	getName: retrieves the first and last name from the DB displays them together on the Home Page.
	getemail: retrieves the email from the DB and displays it on the Home Page.
Referenced By:	1.2.1.2.1
Viewpoint:	Class Diagram

View 1.2.1.2.1.2: Mobile Home Profile Session API Call

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Session",
  "type": "object",
  "properties": {
    "firstName": { "type": "string" },
    "lastName": { "type": "string" },
    "username": { "type": "string" },
    "userID": { "type": "string" },
    "email": { "type": "string", "format": "email" },
    "thumbnailURL": { "type": "string", "format": "uri" }
  },
  "required": ["firstName", "lastName", "username", "userID",
               "email", "thumbnailURL"]
}
```

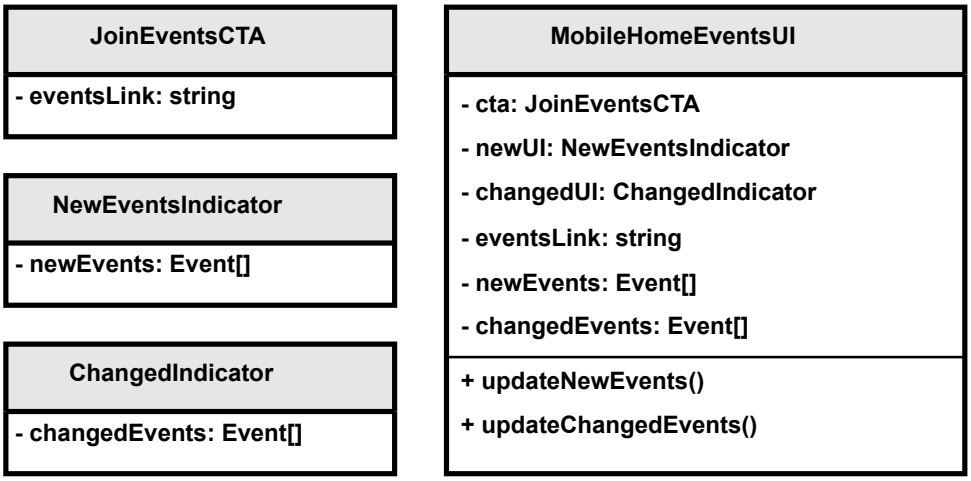
Name	1.2.1.2.1.2 Mobile Home Profile Session API Call
Purpose:	Describes the response format for an API call to retrieve a user's session information
Description:	The Mobile Home Profile UI Component needs to retrieve a user's session information, such as their name, to display within the component.
Requirements:	PO.1.2.12, PO.1.7.17, PO.1.7.21, PO.1.7.34, PO.3.4.2, PO.3.5.2, PO.3.6.2
Elements:	**Various Names:** Various display names connected to the user's account
	User ID: A unique identifier that is used to construct a link to the profile
	Email: The user's email address which can be used elsewhere in the app
	Thumbnail URL: Link to a profile pic thumbnail used in the component
Referenced By:	1.2.1.2.1
Viewpoint:	JSON Schema

View 1.2.1.2.2: Mobile Home Events UI



Name	1.2.1.2.2 Mobile Home Events UI
Purpose:	Provides an area to direct users to join events and alert users to changes to joined events
Description:	A UI component on the mobile version of the homepage which contains different sub-components to provide quick and easy access to common event-related features.
Requirements:	PO.1.7.17, PO.2.1.13, IR.1, IR.4, IR.16
Elements:	**1.0 Join Events CTA:** a prominent UI component containing a link to the main events page where users can browse and join events.
	1.1 New Events Indicator: a UI component which is enabled when there are new events that have not yet appeared in the user's event feed. Displays the number of new events using the Uvents API, and links to the main events page.
	1.2 Changed Events Indicator: a UI component enabled when there are changes to joined events not yet viewed by the user. Links to the individual event page of the changed event retrieved from the Uvents API.
	Uvents API: an external service with endpoints for retrieving changed events and new events data.
Referenced By:	1.2.1.2
Viewpoint:	Component Diagram

View 1.2.1.2.2.1: Mobile Home Events UI Class



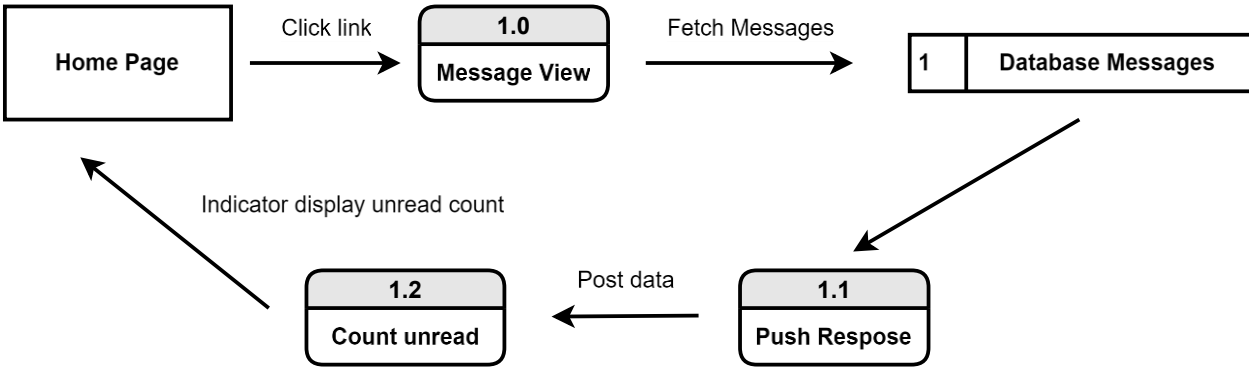
Name	1.2.1.2.2.1 Mobile Home Events UI Classes
Purpose:	The MobileHomeEventsUI class, and the classes of its sub-components, handle the display of the events-related user interface on the mobile home page.
Description:	Each sub-component is responsible for rendering a different portion of the UI, using values relevant to their task passed in from the container component, MobileHomeEventsUI.
Requirements:	PO.1.7.17, PO.2.1.13, IR.1, IR.4, IR.16
Elements:	**MobileHomeEventsUI:** Stores an instance of each sub-component, three in total. In addition, it uses updateNewEvents() and updateChangedEvents() to update the corresponding values for usage by the sub-components.
	JoinEventsCTA: Displays a prominent Call-To-Action on the home page which links to the events feed page using the eventsLink in order to guide users to join events.
	NewEventsIndicator: Uses the list of new events to change its appearance when new events are available, allowing users to tap it to bring up a context menu populated with links to each new event.
	ChangedIndicator: Uses the list of changed events to change its appearance when changed have been made to joined events, allowing users to tap it to bring up a context menu populated with links to each changed event.
Referenced By:	1.2.1.2.2
Viewpoint:	Class Diagram

View 1.2.1.2.2.2: Mobile Home Events UI API Call

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Session",
  "type": "object",
  "properties": {
    "firstName": { "type": "string" },
    "lastName": { "type": "string" },
    "username": { "type": "string" },
    "userID": { "type": "string" },
    "eventName": { "type": "string" },
    "eventID": { "type": "string" },
  },
  "required": ["firstName", "lastName", "username", "userID", "eventName", "eventID"]
}
```

Name	1.2.1.2.2.2 Mobile Home Events UI API Call
Purpose	Describe the API request for event data (new/unseen events)
Description	A schema representing the file used to transfer data requested from the Uvents API by the mobile home events UI component.
Requirements	PO.1.7.17, PO.2.1.13, IR.1, IR.4, IR.16
Elements	Descriptions are included in the schema
Referenced By	1.2.1.2.2
Viewpoint:	JSON Schema

View 1.2.1.2.3: Mobile Home Messages UI



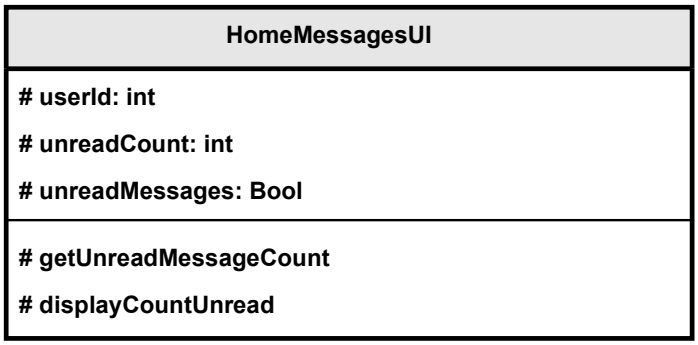
Name	1.2.1.2.3 Mobile Home Messages UI
Purpose	Display flow of Mobile home messages UI
Description	The home navigation includes a message link. The flow indicates where the link leads and how the data is collected from the DB to display an unread message count on the homepage
Requirements	PO. 1.2.12, URS.1.6, URS.1.11
Elements	**Home Page:** the homepage shown to the user.
	1.0 Message view: The page the messages link leads to.
	1 Database Messages Table: source of message data fetched to generate the count of unread messages.
	1.2 Count unread: The response is pushed and processed to produce a count of unread messages. The indicator is displayed on the homepage after login.
Referenced By	1.2.1.2
Viewpoint:	Data flow Diagram

View 1.2.1.2.3.1: Mobile Messages UI API Call

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "newMessages",
  "type": "object",
  "properties": {
    "userid": {
      "type": "string"
    },
    "conversationid": {
      "type": "string"
    },
    "date": {
      "type": "string",
      "format": "date"
    },
    "timestamp": {
      "type": "string",
      "format": "date-time"
    },
    "status": {
      "type": "string"
    }
  },
  "required": ["userid", "conversationid", "date", "timestamp", "status"]
}
```

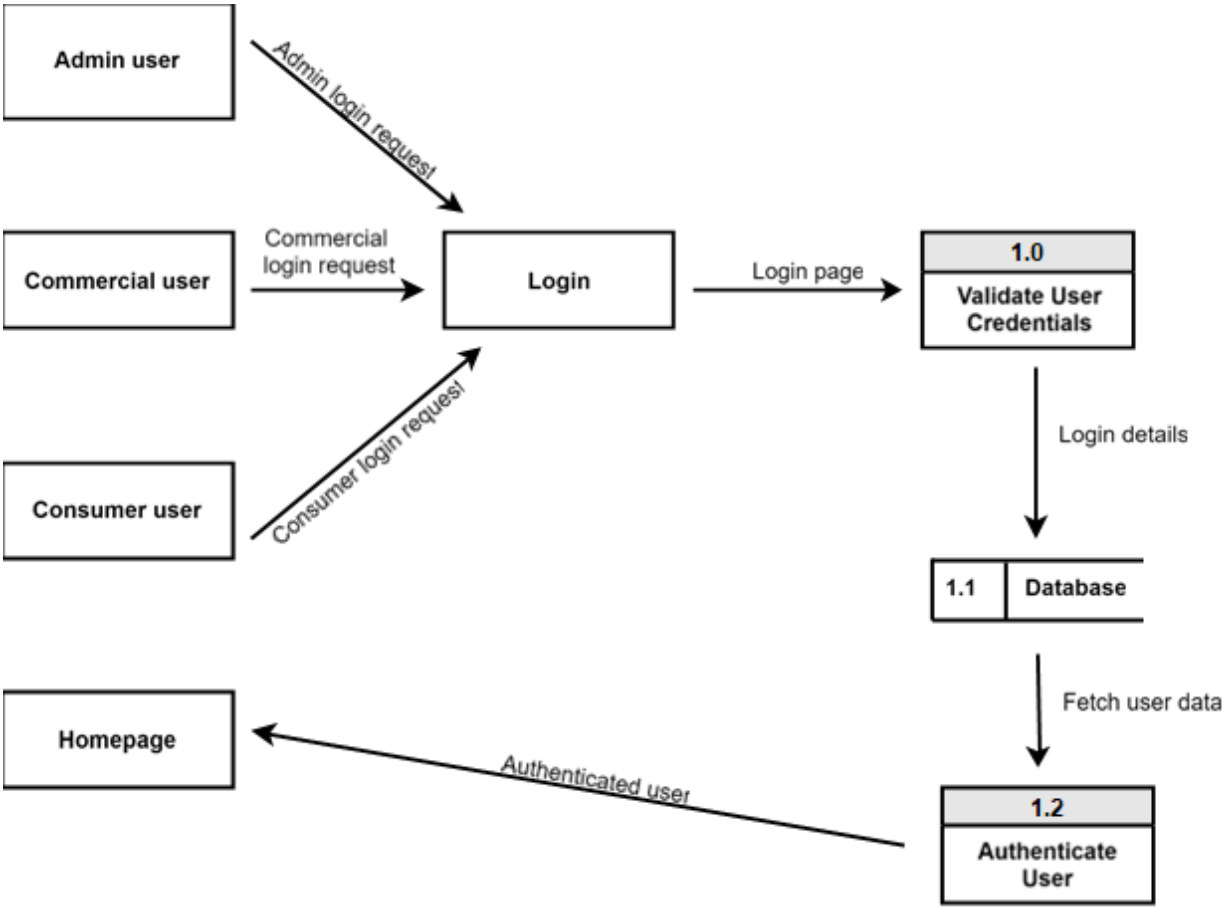
Name	**1.2.1.2.3.1 Mobile Messages UI API Call**
Purpose:	Describe the JSON file representation of the request for unread messages on the home page.
Description:	JSON Schema representing the API request for mobile home unread messages notification
Requirements:	PO. 1.2.12
Elements:	**UserId:** Identifies the user whose messages are being requested.
	Conversation id: identifies which conversation the messages go with.
	Status: shows message status.
Referenced By:	1.2.1.2.3
Viewpoint:	JSON Schema

View 1.2.1.2.3.2: Mobile Home Messages UI Class



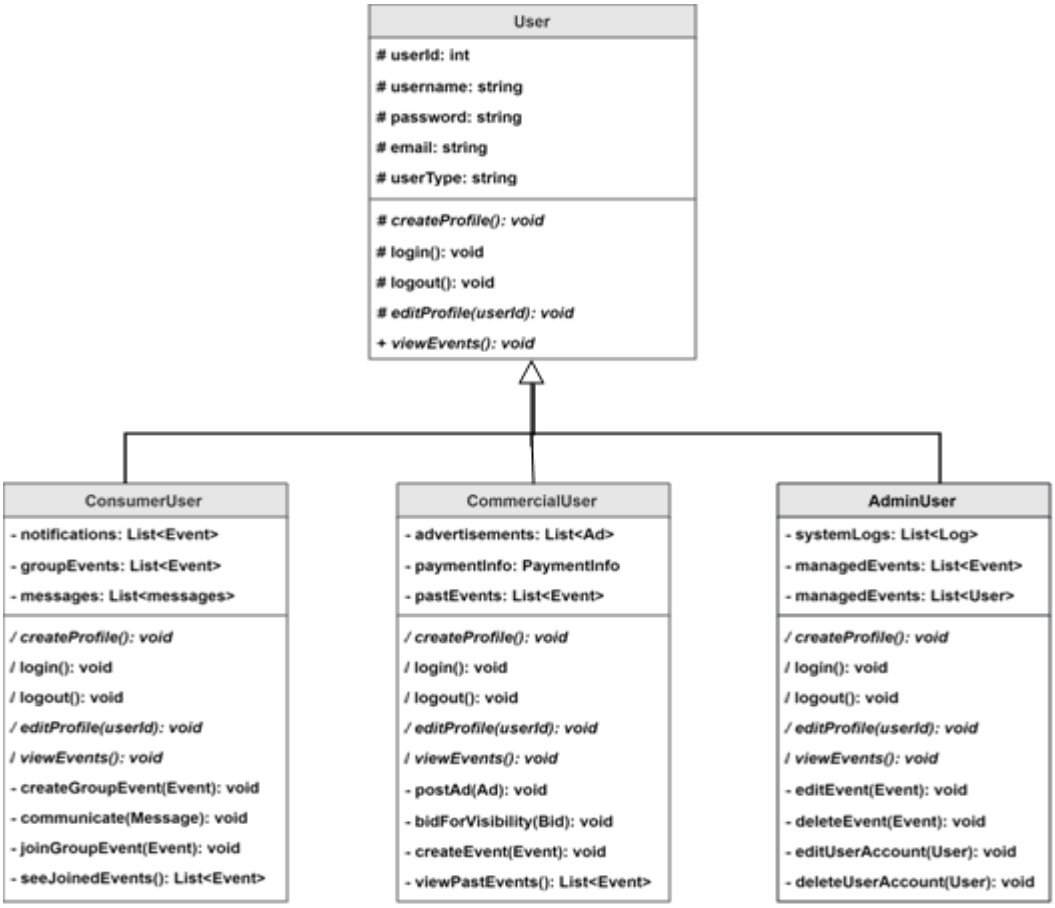
Name	1.2.1.2.3.2 Mobile Home Messages UI Class
Purpose:	Display flow of Mobile home messages UI
Description:	The home navigation includes a message link. The HomeMessagesUI class is used to collect data from the DB to display an unread message count on the homepage.
Requirements:	PO. 1.2.12, URS.1.6, URS.1.11
Elements:	**userId:** Identifies the user
	unreadCount: if unread count is above 0 then the unreadMessages will be set to true. If unreadMessages is true then the displayCountUnread will alter the UI to display the unread count.
	getUnreadMessages: will query the DB to access the unread messages and update the unreadcount variable.
Referenced By:	1.2.1.2.3
Viewpoint:	Class Diagram

View 1.2.1.3: Mobile Log In



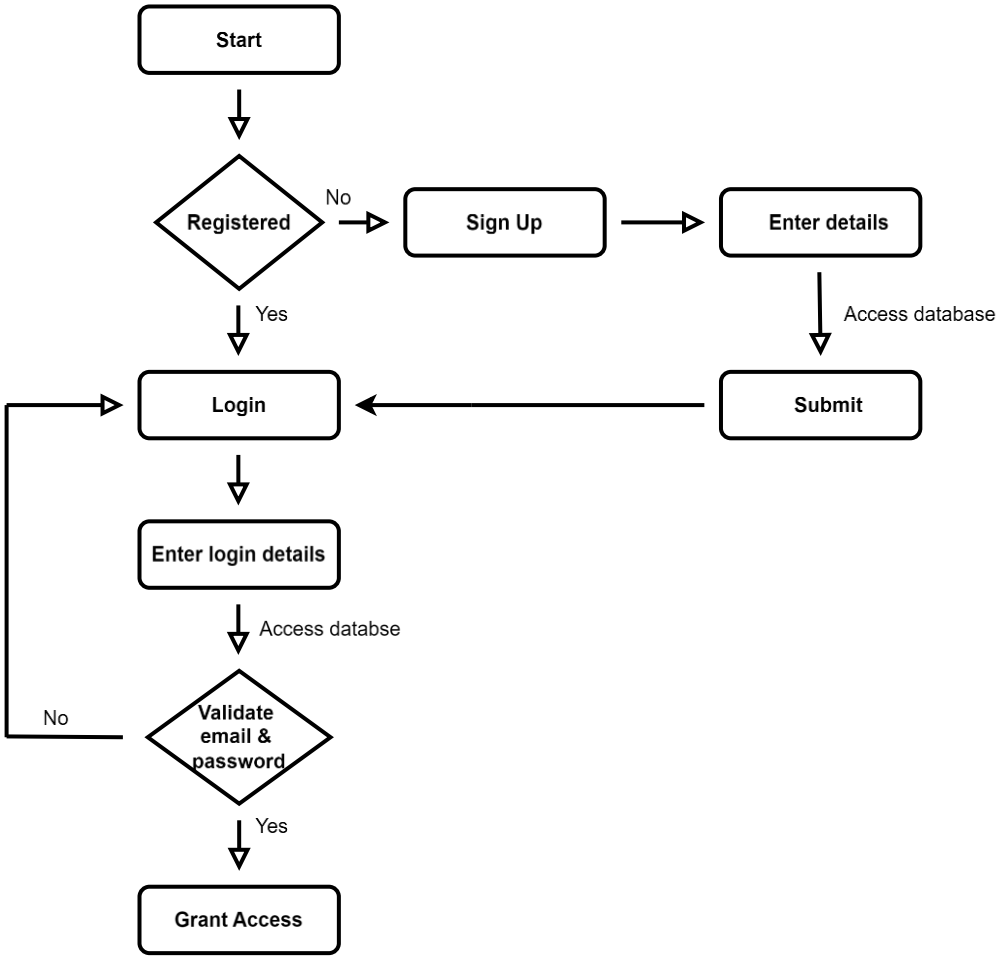
Name	1.2.1.3 Mobile Log In
Purpose:	Provide users with a means of authentication
Description:	The UI for the mobile version of the authentication. Allows users of different status to login to the application.
Requirements:	PO. 1.2.1, PO. 1.2.2
Elements:	**Login**: Brings out the login page associated with the selected user type
	Validate User: Runs frontend validation on the login data
	Authenticate User: Runs server-side validation on the login data
	Homepage: Redirects the user to the homepage on a successful login
Referenced By:	1.2.1.1
Viewpoint:	Data Flow Diagram

View 1.2.1.3.1: Mobile Account User Types



Name	**1.2.1.3.1 Mobile Account User Types**
Purpose:	Illustrate the process of creating a user of each type
Description:	Shows the different parameters that are required for each type of user
Requirements:	PO.1.2.2
Elements:	**User Class:** It is the base class which other subclasses inherit from. It contains the attributes and methods that are the same for all user types.
	Consumer Class: It is a subclass that inherits all the attributes and methods of the User class and contains its own unique attributes and classes specific to the consumer user.
	Commercial Class: It is a subclass that inherits all the attributes and methods of the User class and contains its own unique attributes and classes specific to the commercial user.
	Admin Class: It is a subclass that inherits all the attributes and methods of the User class and contains its own unique attributes and classes specific to the admin user.
Referenced By:	1.2.1.3
Viewpoint:	Class Diagram

View 1.2.1.3.2: Mobile Account Validation



Name	1.2.1.3.2 Mobile Account Validation
Purpose:	Demonstrates the steps involved when a user is trying to login
Description:	The UI for the mobile login displays a form with the email and password fields. It grans the user access to the platform on successful login
Requirements:	FS.15.3, FS.15.6.1, FS.31.2
Elements:	**Start:** A button the user clicks to bring out the login page.
	Sign Up: Brings the sign-up page when clicked by a user from the login view.
	Login: Brings up the login page which contains the login form when the user clicks on the login button.
	Validate Details: Connects to the Uvents database to validate the credentials entered by the user.
	Grant Access: Redirects the user to the homepage after successful login.
Referenced By:	1.2.1.3

Viewpoint:	Flow Chart

View 1.2.1.3.2.1: Login/Oauth API Call

```
{
   “$schema”: "http://json-schema.org/draft-07/schema#",
   “title”: “login”,
   “type”: “object”,
   “properties”: {
      “consumerUser”: {
         “description”: “a user's login info”
         “type”: “object”,
         “properties”: {
            “username”: {
               “description”: “username of the user”
               “type”: “string”
            }
            “socialMediaID”: {
               “description”: “ID of the user”
               “type”: “int”
            }
            “password”: {
               “description”: “password of the user”
               “type”: “string”
            }
            “passwordReset”: {
               “description”: “reset password”
               “type”: “boolean”
            }
            “email”: {
               “description”: “email of the user”
               “type”: “string”
            }
            “role”: {
               “description”: “role of the user”
               “type”: “string”
            }
         }
      }
   }
}
```

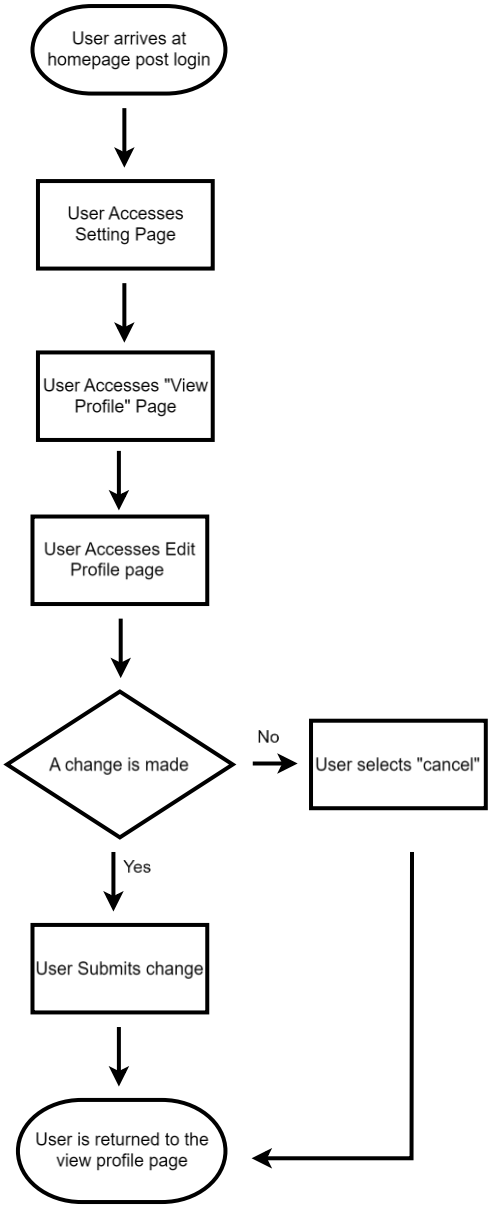
Name	1.2.1.3.2.1: Login/Oauth API Call
Purpose:	Provides JSON data that allows a user to log in to the application
Description:	An API call that will allow a user to log in
Requirements:	PO.1.2.8, PO.1.2.9, PO.1.2.14
Elements:	**1.1.1.1 consumerUser -** Holds the data for an individual user
Referenced By:	1.2.1.3.2
Viewpoint:	JSON Schema

View 1.2.1.3.3: Forgot/Reset Password

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "PasswordReset",
  "type": "object",
  "properties": {
    "userID": {
      "type": "string",
      "description": "The unique identifier of the user requesting password reset."
    },
    "resetToken": {
      "type": "string",
      "description": "Token generated for resetting the user's password."
    },
    "expirationDate": {
      "type": "string",
      "format": "date-time",
      "description": "Expiration date and time of the reset token."
    },
    "email": {
      "type": "string",
      "format": "email",
      "description": "Email address where the password reset link will be sent."
    }
  },
  "required": ["userID", "resetToken", "expirationDate", "email"]
}
```

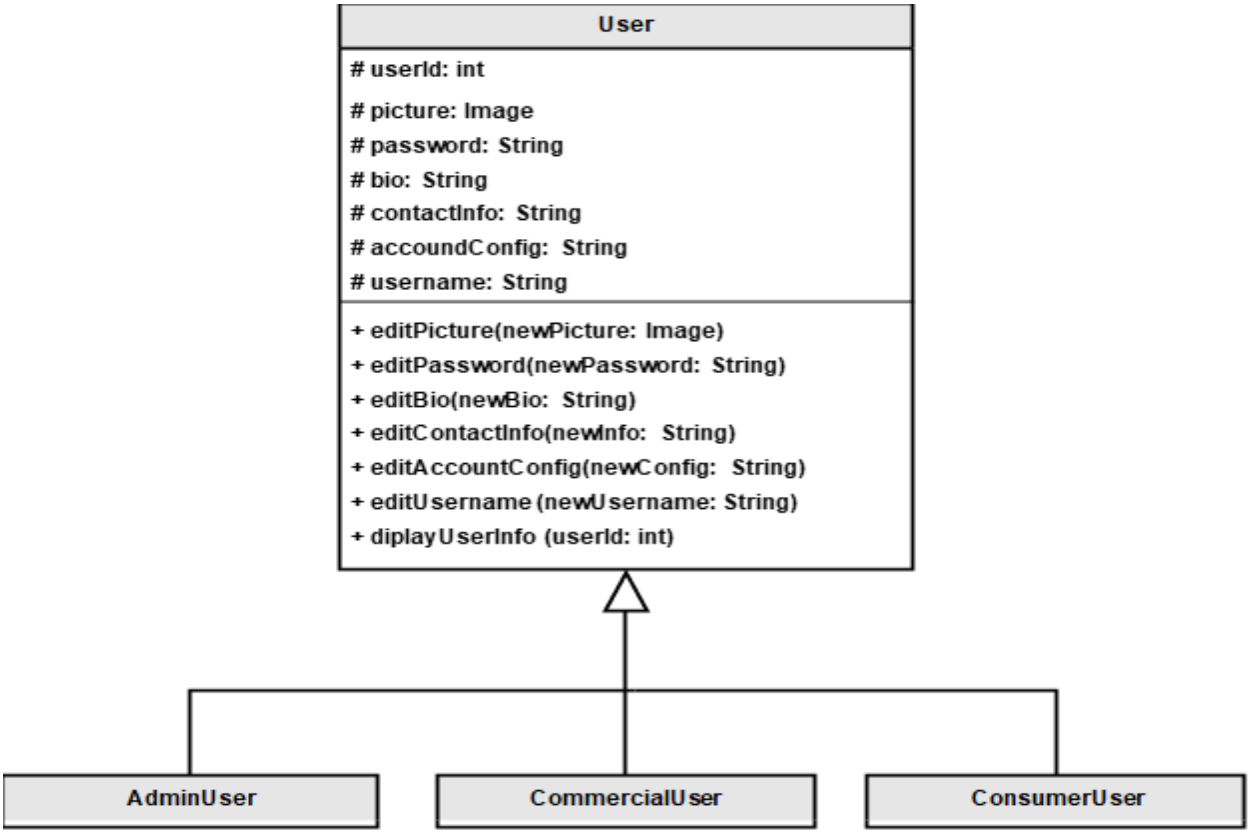
Name	1.2.1.3.3 Forgot/Reset Password
Purpose:	Describe the structure of the data required for enabling the functionality to reset a user's password via email.
Description:	This JSON Schema defines the properties necessary to facilitate the process of sending a password reset email to a user.
Requirements:	LDV.1.11
Elements:	Descriptions are included in the Schema
Referenced By:	1.2.1.3
Viewpoint:	JSON Schema

View 1.2.1.4: Mobile User Profile



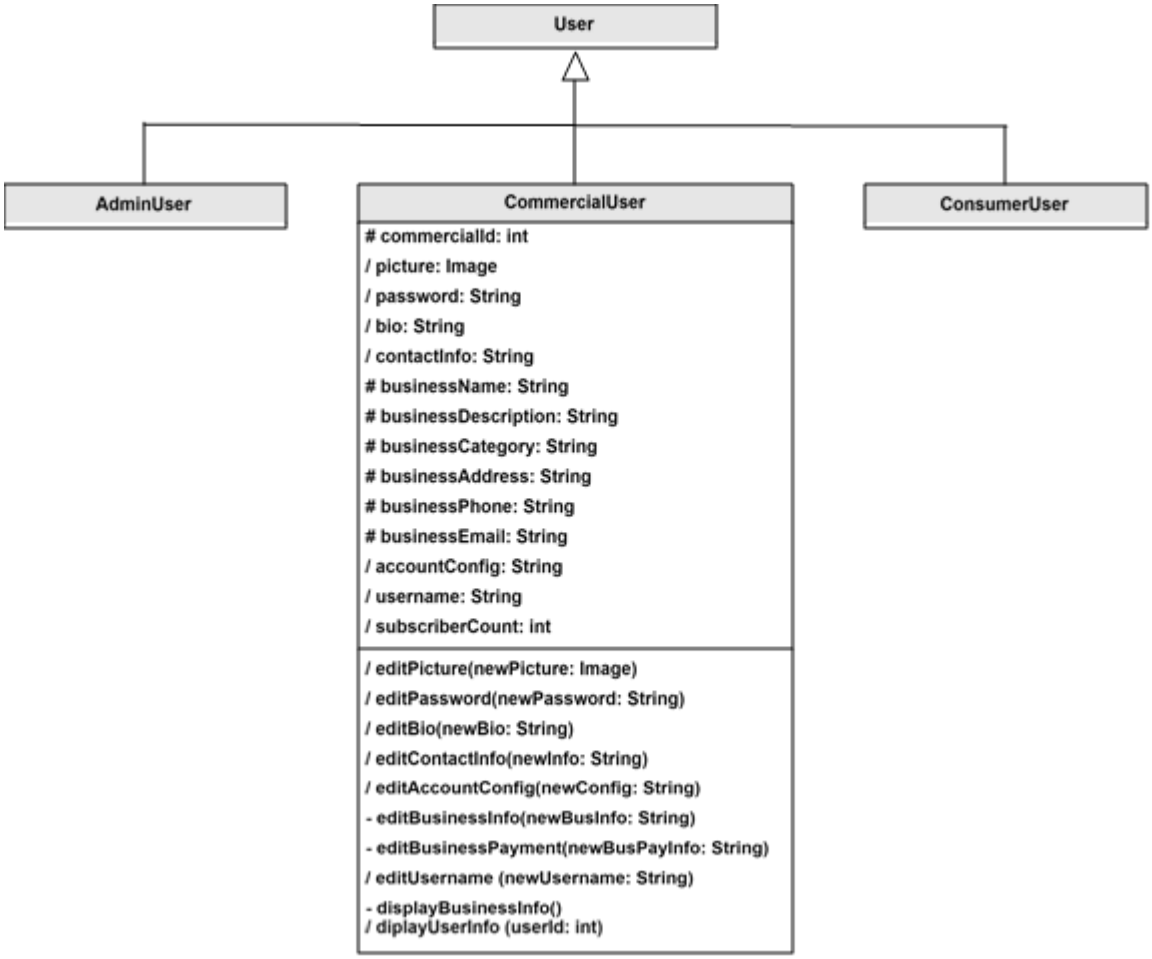
Name:	**1.2.1.4 Mobile User Profile**
Purpose:	Represent the flow of user interaction with the Mobile user profile components.
Description:	From the settings page select View profile. Edit Profile can then be accessed. The user is redirected to the view profile page after submitting or canceling the form
Requirements:	PO.1.2.6
Elements:	**User type:** User logs-in and is directed to a homepage with content based on user type
Referenced By:	1.2.1.1
Viewpoint:	Workflow Diagram

View 1.2.1.4.1: Mobile View and Edit Profile



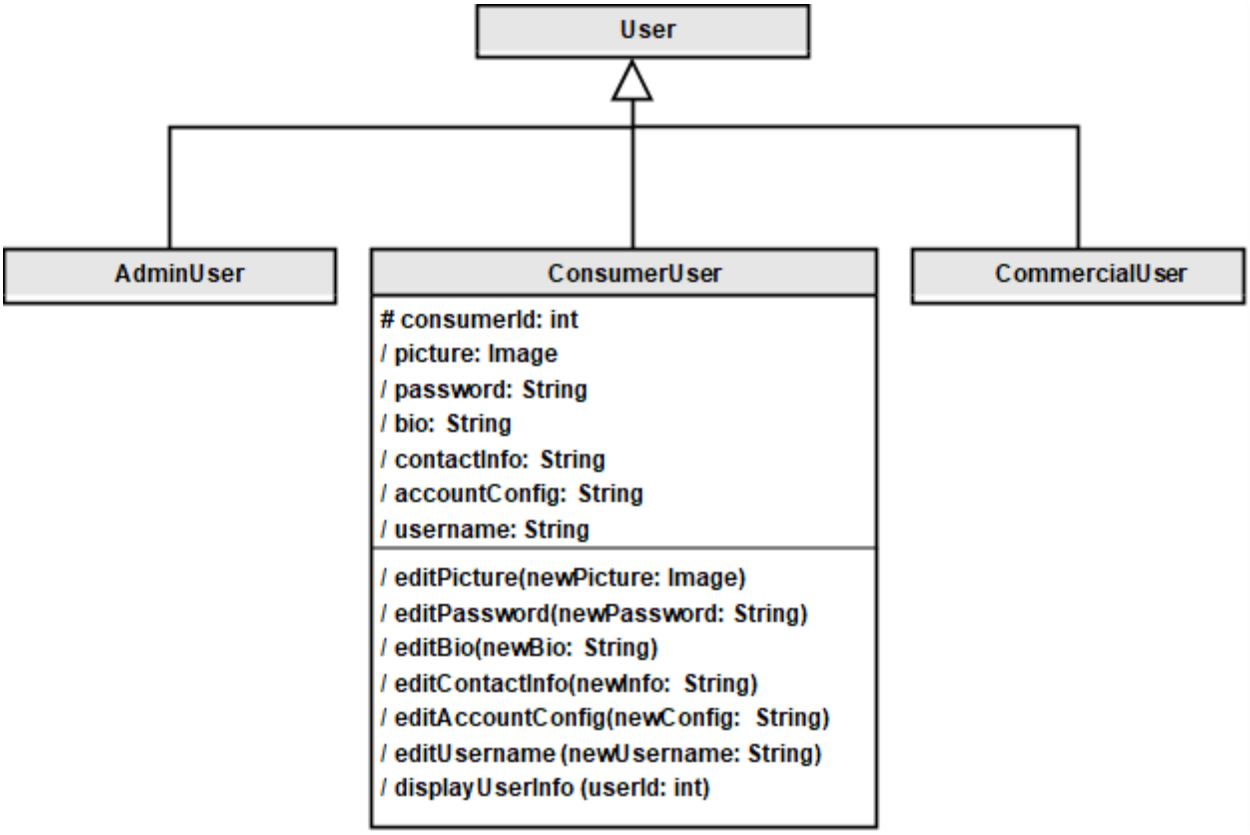
Name	1.2.1.4.1 Mobile View and Edit Profile
Purpose:	Define the classes and function involved in using the view and edit profile feature for the different user types.
Description:	A class diagram showing the breakdown of the three user types into classes which inherit variables and methods from the User class.
Requirements:	PO.1.2.6, PO.2.1.4, PO.2.2.5, PO.2.2.6, PO.2.2.7, PO.2.3.3
Elements:	**1.2.1.4.1.1 Commercial User:** A class which inherits the User class's functions used by commercial users.
	1.2.1.4.1.2 Consumer User: A class which inherits functions and variables from the User Class
	1.2.1.4.1.3 Admin User: Admin class inherits variables and methods from the user class
	User: A class containing the functions accessed by the editProfile functions of the different user type classes
Referenced By:	1.2.1.4
Viewpoint:	Class Diagram

View 1.2.1.4.1.1 Commercial User Class



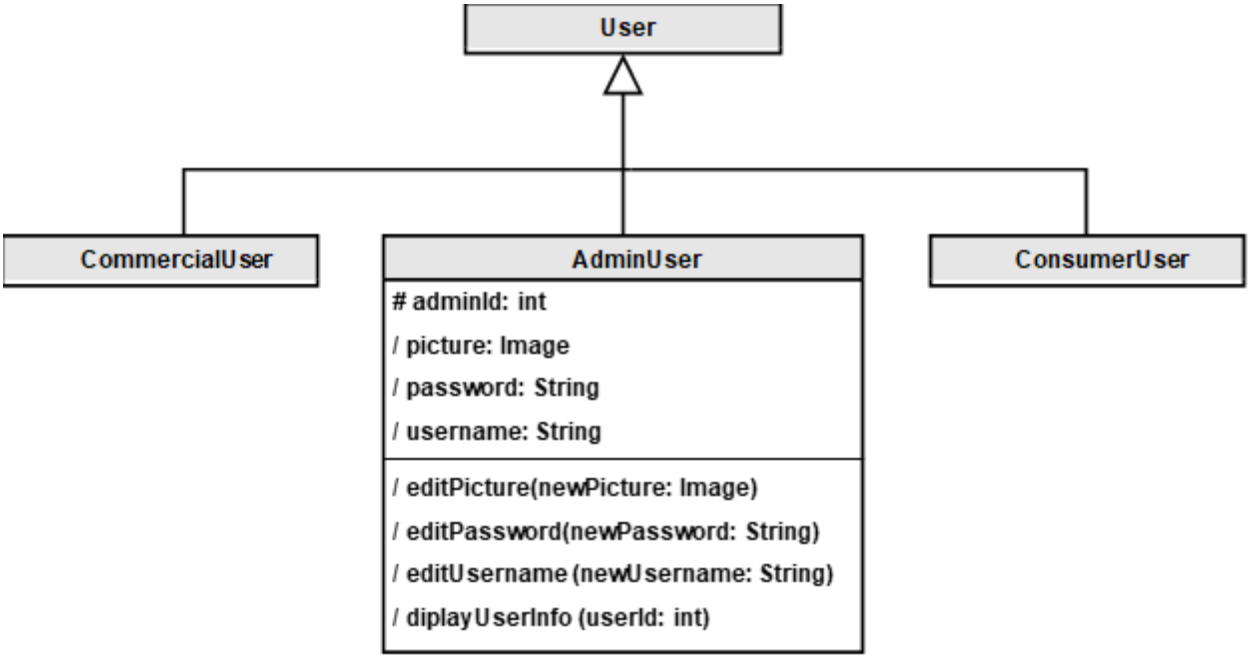
Name:	**1.2.1.4.1.1 Commercial User Class**
Purpose:	Define the classes and function involved in using the view and edit profile feature for the for commercial Users
Description:	A class diagram showing the breakdown of the three user types into classes which inherit variables and methods from the User class.
Requirements:	PO.1.2.6, PO.2.1.4, PO.2.2.5, PO.2.2.6, PO.2.2.7, PO.2.3.3
Elements:	**Commercial User:** A class which inherits the User class's functions used by commercial users.
	Business Variables and Functions: These are specific to commercial users and are not inherited from the User class.
Referenced By:	1.2.1.4.1
Viewpoint:	Class Diagram

View 1.2.1.4.1.2: Consumer User Class



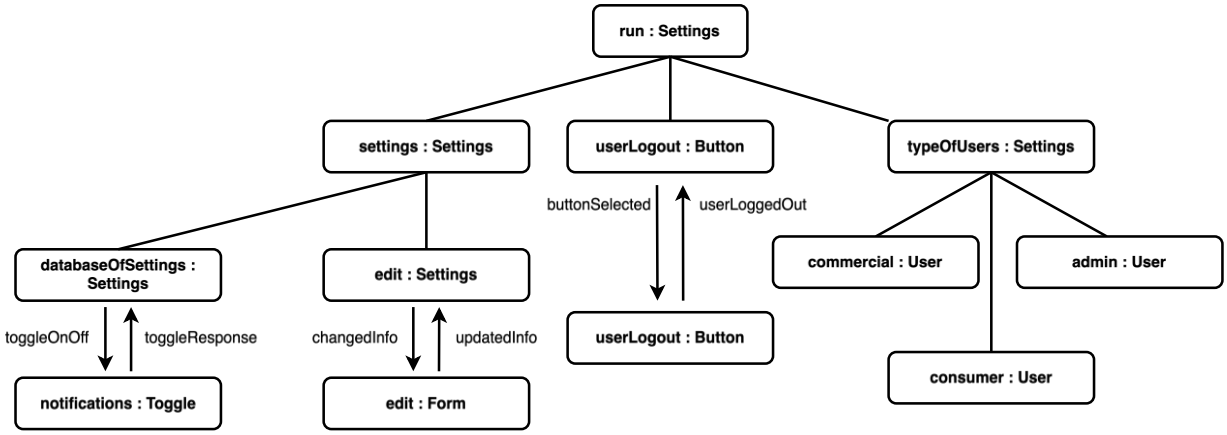
Name	1.2.1.4.1.2 Consumer User Class
Purpose:	Define the classes and function involved in using the view and edit profile feature for the for consumer Users
Description:	A class diagram showing the breakdown of the three user types into classes which inherit variables and methods from the User class.
Requirements:	PO.1.2.6, PO.2.1.4, PO.2.2.5, PO.2.2.6, PO.2.2.7, PO.2.3.3
Elements:	**Consumer User:** A class which inherits the User class's functions used by consumer users.
	consumerId: A variable identifying a consumer
Referenced By:	1.2.1.4.1, 1.2.1.4.1.1, 1.2.1.4.1.3
Viewpoint:	Class Diagram

View 1.2.1.4.1.3: Admin User Class



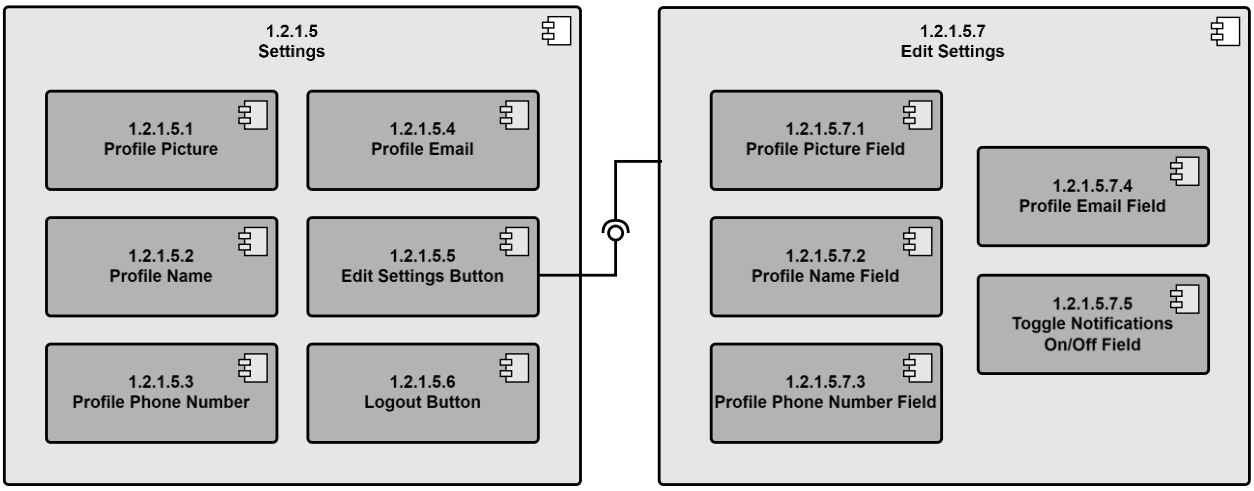
Name	1.2.1.4.1.3 Admin User Class
Purpose:	Define the classes and function involved in using the view and edit profile feature for the for admin Users
Description:	A class diagram showing the breakdown of the three user types into classes which inherit variables and methods from the User class.
Requirements:	PO.1.2.6, PO.2.1.4, PO.2.2.5, PO.2.2.6, PO.2.2.7, PO.2.3.3
Elements:	**Admin User:** A class which inherits the User class's functions used by admin users.
	adminId: A variable identifying an admin user
Referenced By:	1.2.1.4.1, 1.2.1.4.1.1, 1.2.1.4.1.2
Viewpoint:	Class Diagram

View 1.2.1.5: Mobile Settings



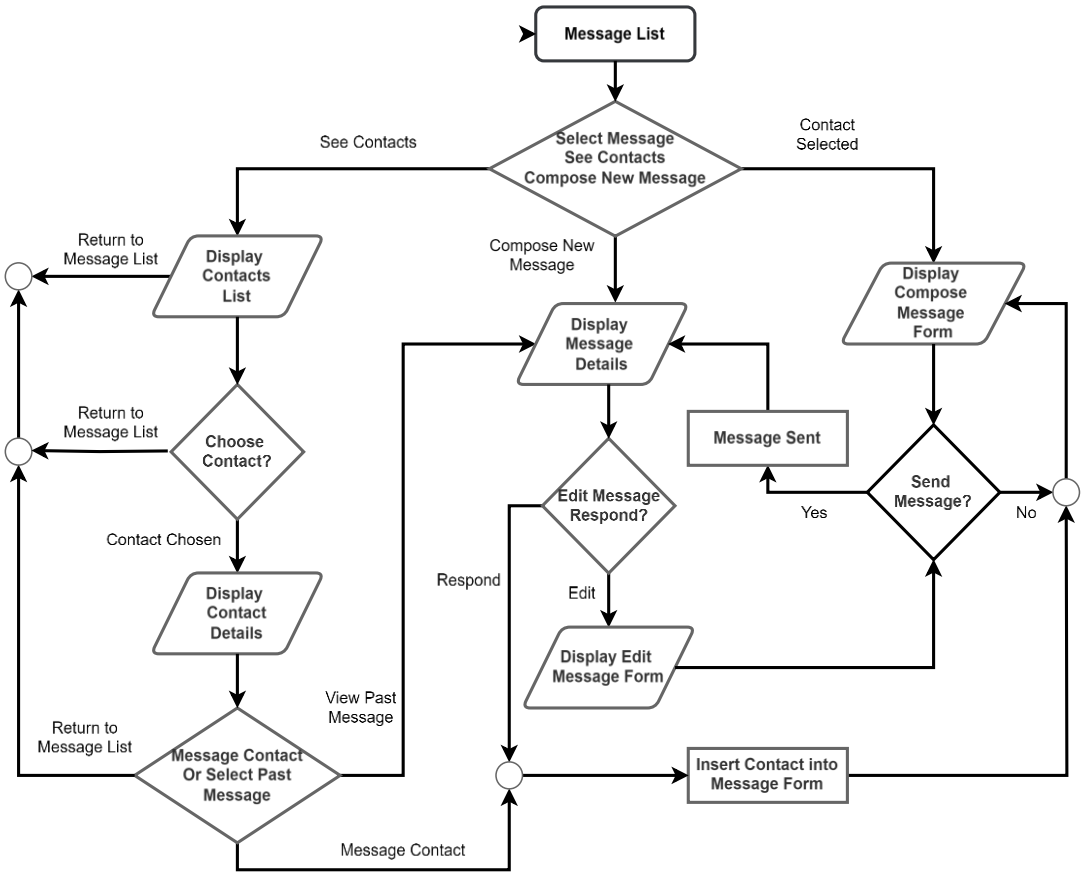
Name	1.2.1.5: Mobile Settings
Purpose:	To allow users to view and edit settings as well as logout
Description:	The mobile settings view is designed to allow users to easily view all their settings. It also gives the ability to edit settings, which are pulled from a database of all the different settings that can be edited. One of those settings that has already been identified is to be able to toggle notifications on or off. Also, this would allow users to log out of the system as well.
Requirements:	PO. 1.2.3, PO. 1.2.11
Elements:	**Settings**: Brings up the Settings page associated with the selected user type
	Edit Settings: Brings up the Settings options that can be edited or changed
	Database of Settings: Is a database of options that can be changed based on the selected user type
	Toggle Notification On/Off: Is a specific setting that toggles notifications on or off for the selected user
	Logout: Gives the user the ability to log out of the system from the setting page
Referenced By:	1.2.1.1
Viewpoint:	Structure Chart

View 1.2.1.5.1: Mobile Additional Settings Options



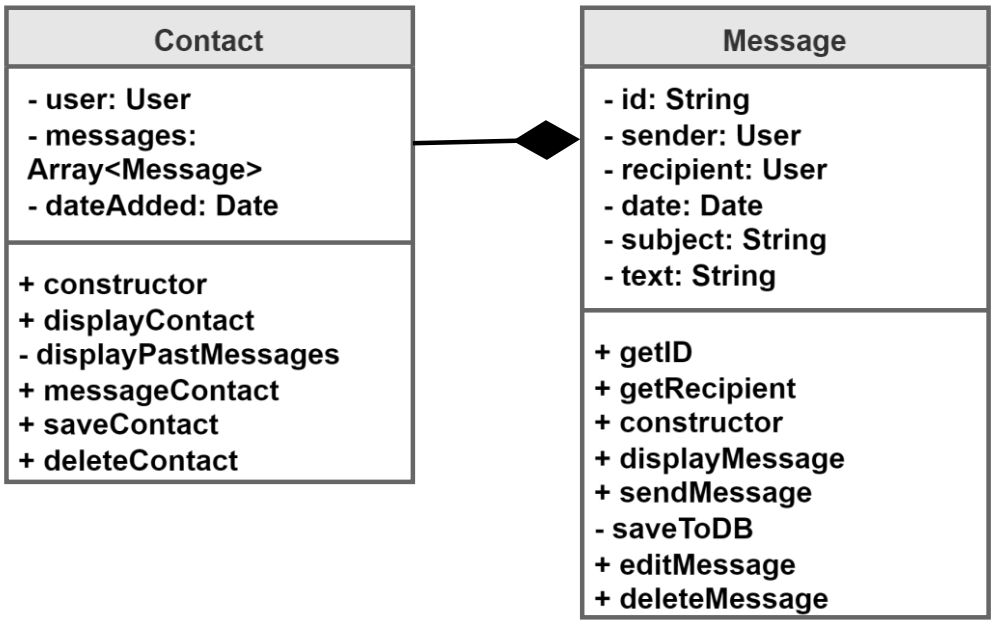
Name	1.2.1.5.1 Mobile Additional Settings Options
Purpose:	To allow users to view and edit settings as well as logout.
Description:	The mobile settings view is designed to allow users to easily view all their settings. It also gives the ability to edit those settings and allow users to log out of the system as well.
Requirements:	PO.1.2.3, PO.1.2.11
Elements:	1.2.1.5 **Settings:** Displays all the Settings.
	1.2.1.5.1 **Profile Picture:** The user's picture is displayed
	1.2.1.5.2 Profile Name: The user's name is displayed.
	1.2.1.5.3 Profile Phone Number: The user's phone number is displayed
	1.2.1.5.4 Profile Email: The user's email is displayed
	1.2.1.5.5 Edit Settings Button: A button to open the Edit Settings page
	1.2.1.5.6 Logout Button: A button for the user to log out of the system
	1.2.1.5.7 Edit Settings: Brings up the page to edit the settings
	1.2.1.5.7.1 Profile Picture Field: Field to edit user's picture
	1.2.1.5.7.2 Profile Name Field: Field to edit user's name
	1.2.1.5.7.3 Profile Phone Number Field: Field to edit phone number
	1.2.1.5.7.4 Profile Email Field: Field to edit user's profile email
	1.2.1.5.7.5 Toggle Notifications On/Off Field: Setting to toggle notifications on or off for the selected user
Referenced By:	1.2.1.1
Viewpoint:	Component Diagram

View 1.2.1.6: Mobile Messaging



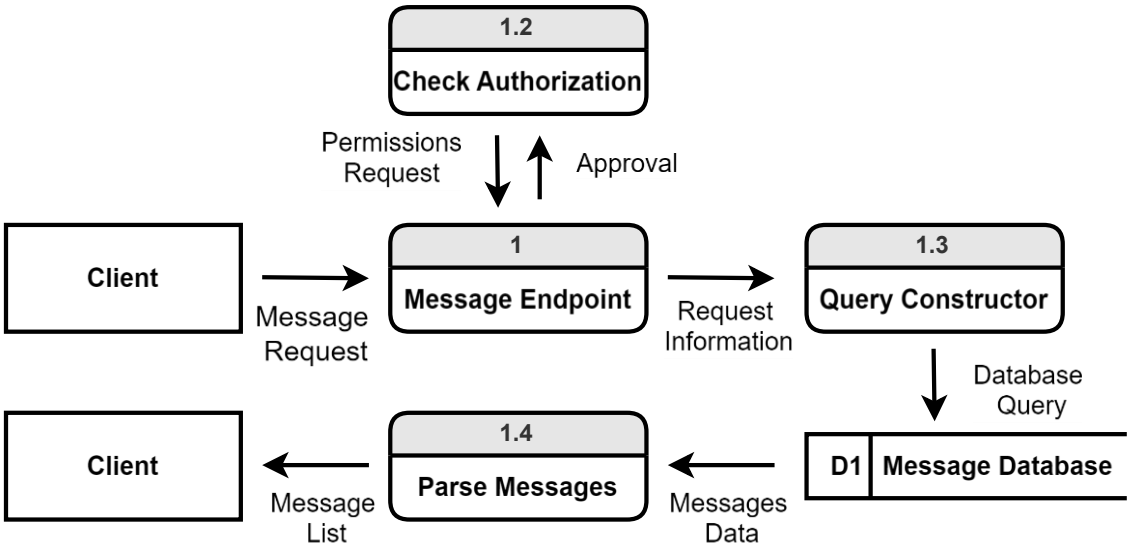
Name	1.2.1.6 Mobile Messaging
Purpose:	To show how a user is able to move through the messaging app
Description:	The UI for the mobile messaging component displays the user's contacts, their messages, and allows them to edit or create new messages.
Requirements:	PO. 1.2.5
Elements:	**Message List**: A scrollable list that displays a user's previous messages in chronological order.
	Message Details: Displays a message in its entirety to the user, as well as any replied messages the displayed message might have received.
	Edit Message: Allows users to change or delete a previously sent message. The message content is put in a form to facilitate changes.
	Compose Message: Presents a form to message other users on the Uvents app. It requires a recipient and text for a message that can be sent after the submit button is pressed.
	1.2.1.6.1.2 Contacts: A scrollable list that displays the user's contacts.
Referenced By:	1.2.1.1, 1.2.1.6.1
Viewpoint:	Flow Chart

View 1.2.1.6.1 Mobile Messaging Class View



Name	1.2.1.6.1 Mobile Messaging Class View
Purpose:	To hold the attributes and methods of messages exchanged between users.
Description:	The Contact class contains information on other users that the active user has saved, as well as the message history between them.
Requirements:	PO. 1.2.5, PO.1.7.7, PO1.7.23, PO.1.7.36
Elements:	**Message Class:** holds the attributes and methods of messages exchanged between users including sender and recipient information, the date sent, the subject, and the text content. It includes methods to display, send, edit, delete, and save messages to a database.
	Contact Class: The Contact class stores information about other users saved by the active user, along with the history of messages exchanged with them. It provides methods to display contact details, view past messages, initiate new messages, and manage the contact list, including adding, saving, and deleting contacts.
Referenced By:	1.2.1.6
Viewpoint:	Class Diagram

View 1.2.1.6.1.1: Message API Call



Name	View 1.2.1.6.1.1 Message API Call
Purpose:	To illustrate how data moves throughout the app when the previous messages are requested.
Description:	This view depicts the process of retrieving messages from the message database and the flow of data involved.
Requirements:	PO.2.1.12
Elements:	**Client**
	Messages Endpoint 1: Receives the message request and coordinates the subsequent actions.
	Authorization Function 1.2: Checks to see if the user has the necessary permissions to access the messages.
	Query Constructor Function 1.3: Constructs queries based off the messages that have been requested
	Message Database
	Parse Messages Function 1.4: Parses the message data retrieved from the database into a list of message classed objects.
Referenced By:	1.2.1.6, 1.2.1.6.1
Viewpoint:	Data Flow Diagram

View 1.2.1.6.1.1.1: Message Query SQL

```
SELECT
        m.messageID,
        m.originator AS sender
        c.userID AS recipient,
        m.content AS text,
        m.date,
        m.status
FROM
      message m
      JOIN conversation c ON m.conversationID = c.conversationID
      JOIN consumerUser u1 ON m.originator = u1.userID
      JOIN consumerUser u2 ON c.userID = u2.userID
WHERE
      c.userID = ?
ORDER BY
        m.date DESC,
        m.time DESC;
```

Name	**1.2.1.6.1.1.1 Message Query SQL**
Purpose	This view illustrates what the Query Constructor class will produce when retrieving past messages from the Message database.
Description	This query retrieves the message information required for filling out the Message Class from the Message and Conversation database.
Requirements	PO.1.2.5
Elements	**Message Table 1.1.1.10:** This table stores the messageID, sender, text, date, and status needed for filling out the message class.
	Conversation Table 1.1.1.10: This table holds the recipient(s) information needed for filling out the message class.
Referenced By	1.2.1.6.1.1
Viewpoint	Pseudocode

View 1.2.1.6.1.2: Contact List API Call

```
{
  “$schema”: “http://json-schema.org/draft-07/schema#”,
  “title”: “ContactList”,
  “type”: “object”,
  “properties”: {
   “contacts”: {
    “type”: “array”,
    “items”: {
     “type”: “object”,
     “properties”: {
      “userID”: { “type”: “string” },
      “username”: { “type”: “string” },
      “profilePic”: { “type”: “string”, "format": "uri" },
      “profileLink”: { “type”: “string”, "format": "uri" },
      },
      “required”: [“userID”, “username”, “profileLink”]
     }
    }
   }
}
```

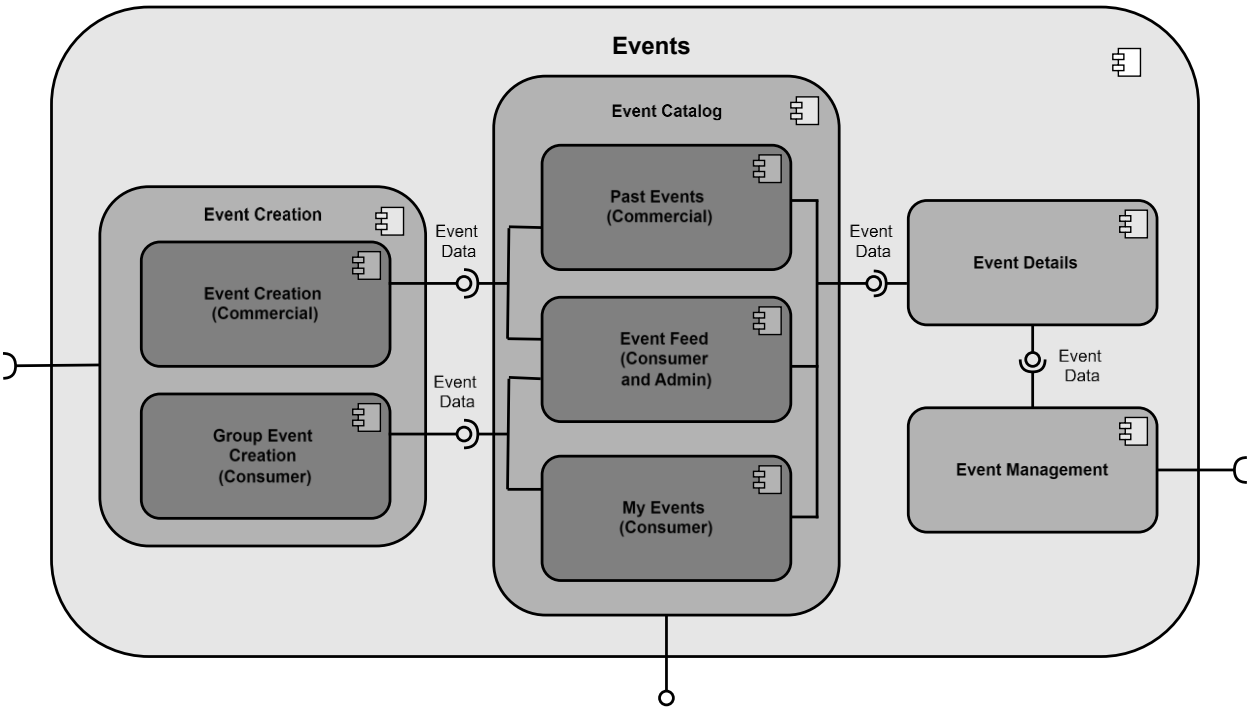
Name	1.2.1.6.1.2 Contact List API Call
Purpose:	Show the format in which contact information is sent to and retrieved from the database
Description:	An API call to get a user’s contact list and the details pertaining to each contact
Requirements:	PO.1.2.5, PO.1.7.7, PO.1.7.23, PO.1.7.36
Elements:	**1.2.1.6.1.2 Contacts:** User’s contact list
	1.2.1.6.1 Contact: Each entry in the contact list, representing a single contact
Referenced By:	1.2.1.6.1
Viewpoint:	JSON Schema

View 1.2.1.6.1.3: Send Message API Call

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "message",
  "type": "object",
  "properties": {
    "userid": {
      "type": "string",
      "description": "The unique identifier for the user."
    },
    "messageid": {
      "type": "string",
      "description": "The unique identifier for the message."
    },
    "sender": {
      "type": "string",
      "description": "The unique identifier for the sender."
    },
    "recipient": {
      "type": "string",
      "description": "The unique identifier for the recipient."
    },
    "date": {
      "type": "string",
      "format": "date-time",
      "description": "The date the message was sent."
    },
    "subject": {
      "type": "string",
      "description": "The subject of the message."
    },
    "text": {
      "type": "string",
      "description": "The text of the message."
    }
  },
  "required": ["userid", "messageid", "sender", "recipient", "date", "subject", "text"]
}
```

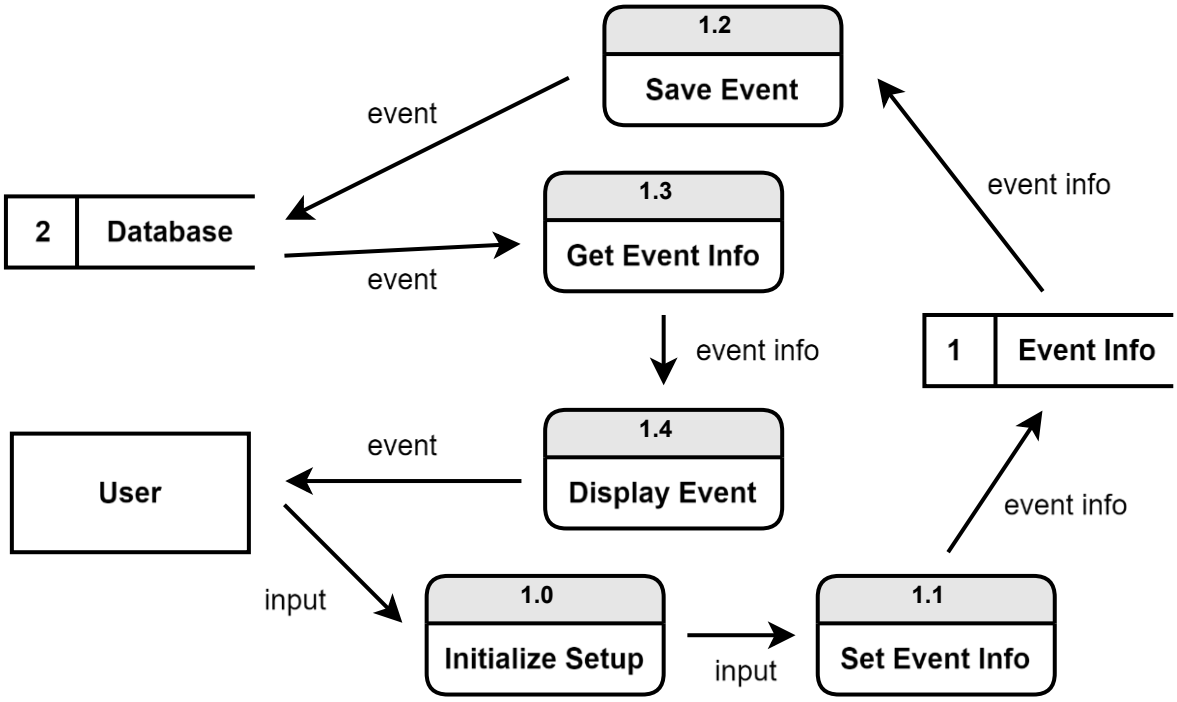
Name	1.2.1.6.1.3 Send Message API Call
Purpose:	Describe the JSON file representation of the data sent on the send message request
Description:	JSON Schema representing the API request for sending a message
Requirements:	PO.2.1.12 PO.2.3.9
Elements:	Descriptions are included in the Schema
Referenced By:	1.2.1.6.1
Viewpoint:	JSON Schema

View 1.2.1.7 Mobile Events Overview



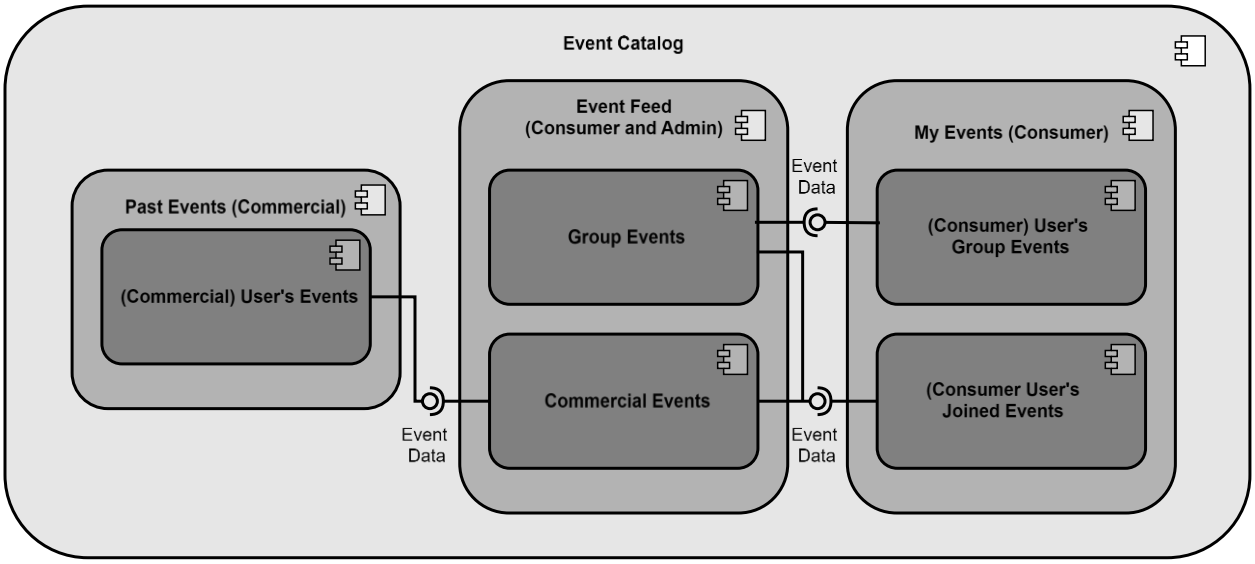
Name	1.2.1.7 Mobile Events Overview
Purpose:	Show how all event components are connected
Description:	The Events Overview shows the fundamental elements of the events page. It covers the creation of events for various users, event list views, individual event details display, and event management options.
Requirements:	PO. 1.2.4, PO. 1.2.7, PO. 1.2.13, PO. 1.2.14
Elements:	1.2.1.7.1 Event Creation
	1.2.1.7.1.2.1 Event Creation (Commercial)
	1.2.1.7.1.2.2 Group Event Creation (Consumer)
	1.2.1.7.2 Event Catalog: Outlines the event list displays for each user
	1.2.1.7.2.1 Past Events (Commercial)
	1.2.1.7.2.2 Event Feed (Consumer and Admin)
	1.2.1.7.2.3 My Events (Consumer)
	1.2.1.7.3 Event Details: Additional information about the details pertaining to a singular event
	1.2.1.7.4 Event Management: Edit, copy, and delete events
	1.2.1.7.5 Event Data
Referenced By:	1.2.1.1
Viewpoint:	Component Diagram

View 1.2.1.7.1: Mobile Event Creation



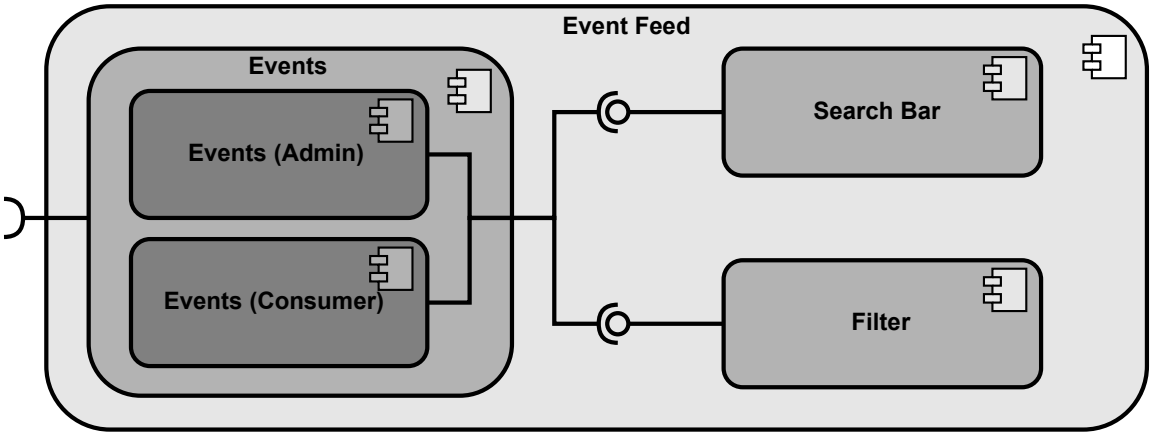
Name	1.2.1.7.1 Mobile Event Creation
Purpose:	Allows users to create events
Description:	Mobile Event Creation allows users to create events, tailored to each user type, which includes entering detailed event information, saving the event to the database, then retrieving the event details for further actions.
Requirements:	PO. 1.2.4, PO. 1.7.8, PO. 1.7.25
Elements:	**1.2.1.7.1.1 Initialize Setup**
	1.2.1.7.1.2 Set Event Info
	1.2.1.7.1.3 Save Event
	1.2.1.7.1.4 Get Event Info
	1.2.1.7.3.1 Display Event
Referenced By:	1.2.1.7
Viewpoint:	Data Flow Diagram

1.2.1.7.2 Mobile Event Catalog



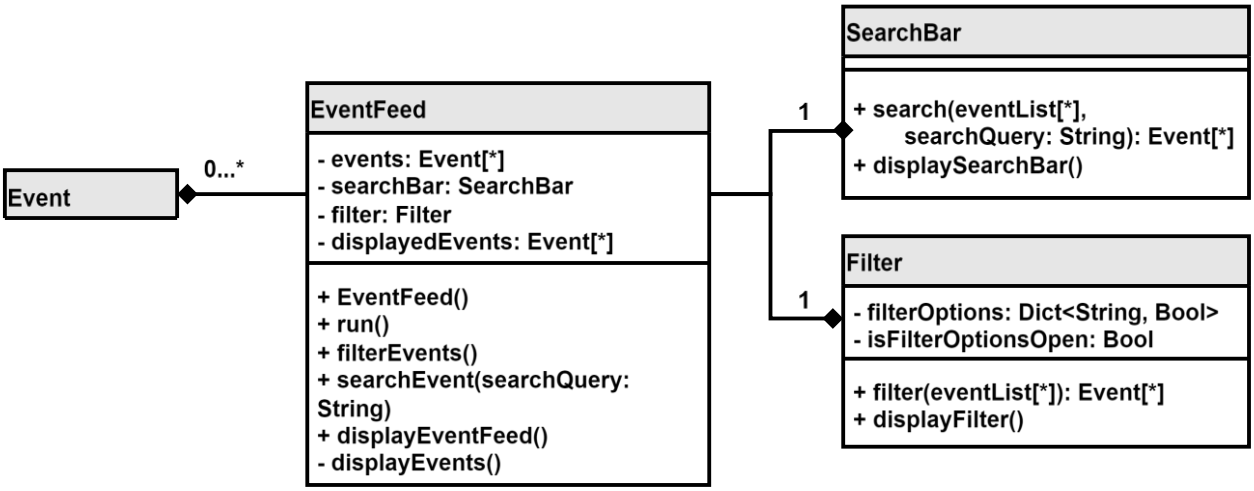
Name	1.2.1.7.2 Event Catalog
Purpose:	Outline the event list displays for each user and how they are connected
Description:	The Event Catalog outlines the various displays of event lists tailored to each user and shows the data connections within the application. It walks through past events, my events, and event feed.
Requirements:	PO.1.2.13, PO.1.2.14, PO.1.7.6, PO.1.7.22, PO.2.1.13, PO.2.2.14
Elements:	**1.2.1.7.2.1 Past Events (Commercial):** Shows commercial users all the events they have previously created
	1.2.1.7.2.1.1 (Commercial) User's Events: Events created by a commercial user
	1.2.1.7.2.2 Event Feed (Consumer and Admin): Displays all group and commercial events for both consumers and admin to view
	1.2.1.7.2.2.3 Commercial Events
	1.2.1.7.2.2.4 Group Events
	1.2.1.7.2.3 My Events (Consumer): Shows consumer users all the events they have both created and joined
	1.2.1.7.2.3.1 (Consumer) User's Group Events: Group events created by a consumer user
	1.2.1.7.2.3.2 (Consumer) User's Joined Events: Events (both commercial and group) joined by a consumer user
	1.2.1.7.5 Event Data
Referenced By:	1.2.1.7
Viewpoint:	Component Diagram

View 1.2.1.7.2.2: Mobile Event Feed



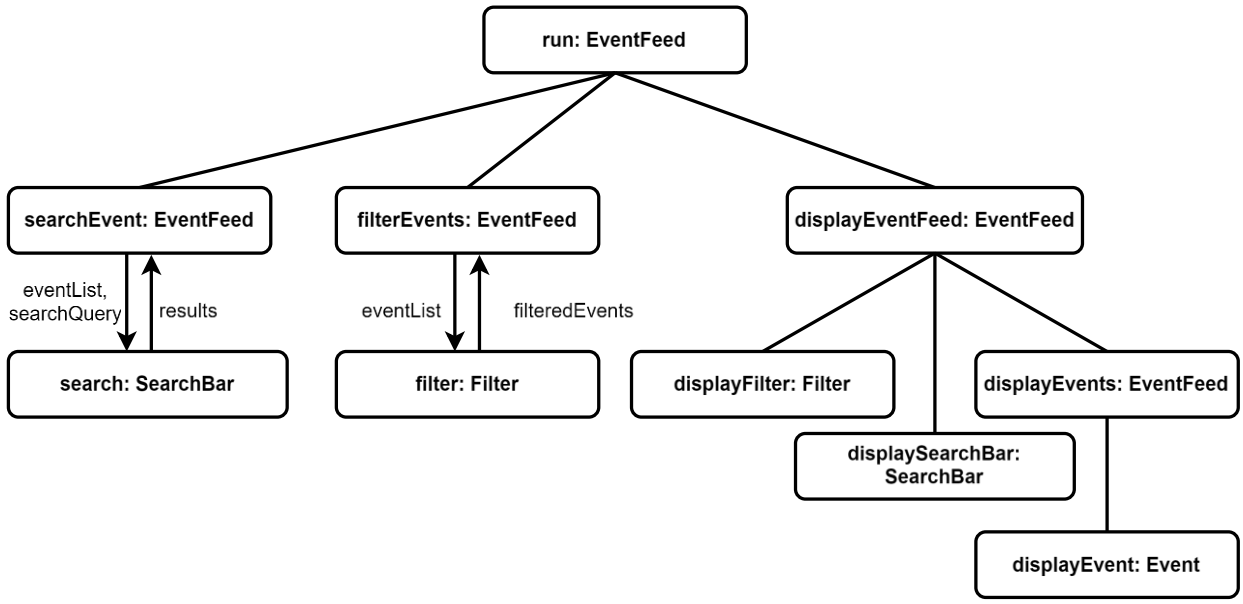
Name	1.2.1.7.2.2 Mobile Event Feed
Purpose:	Provide users a way to view, search for, and filter a list of events
Description:	The UI for the mobile version of the event feed. It displays a list of events that can change based on selected filters and/or search queries
Requirements:	PO. 1.2.8, PO. 1.2.9, PO. 1.2.14
Elements:	1.2.1.7.2.2.3 **Events**: A list of events with relevant event information with two different version (Events (Admin) and Events (Consumer))
	1.2.1.7.2.2.3.1 Events (Admin): A list of events only visible to admin accounts
	1.2.1.7.2.2.3.2 Events (Consumer): A list of events visible to consumer accounts
	1.2.1.7.2.2.4 Search Bar: Allows for a string to be entered to reduce the number of events that are presented based on the search query
	1.2.1.7.2.2.5 Filter: Contains a list of options that will reduce the number of events that are presented to only events that match the selected filtering options
Referenced By:	1.2.1.7, 1.2.1.7.2
Viewpoint:	Component Diagram

View 1.2.1.7.2.2.1 : Mobile Event Feed (Class Diagram)



Name	1.2.1.7.2.2.1 Mobile Event Feed
Purpose:	Provide users a way to view, search for, and filter a list of events
Description:	The UI for the mobile version of the event feed. It displays a list of events that can change based on selected filters and/or search queries
Requirements:	PO.1.2.8, PO.1.2.9, PO.1.2.14
Elements:	**1.2.1.7.2.2.1.1 Event Feed**
	1.2.1.7.2.2.1.2 Search Bar: Allows for a string to be entered to reduce the number of events that are presented based on the search query
	1.2.1.7.2.2.1.3 Filter: Contains a list of options that will reduce the number of events that are presented to only events that match the selected filtering options
	1.2.1.7.3 Event
Referenced By:	1.2.1.7, 1.2.1.7.2
Viewpoint:	UML Class Diagram

View 1.2.1.7.2.2.2 Mobile Event Feed (Structure Chart)



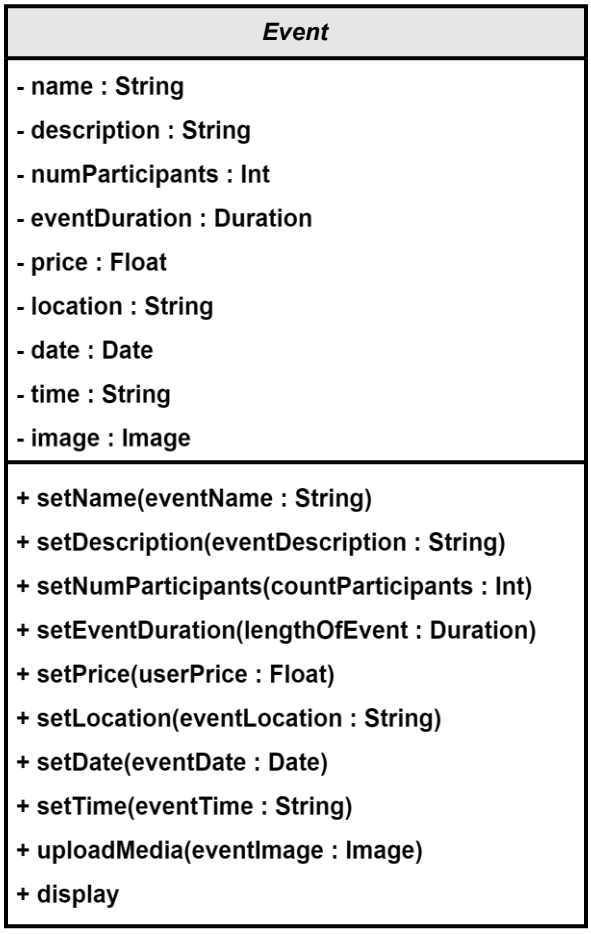
Name	**1.2.1.7.2.2.2 Mobile Event Feed**
Purpose:	Provide users a way to view, search for, and filter a list of events
Description:	The UI for the mobile version of the event feed. It displays a list of events that can change based on selected filters and/or search queries
Requirements:	PO.1.2.8, PO.1.2.9, PO.1.2.14
Elements:	**1.2.1.7.2.2.2.1 run: EventFeed**
	1.2.1.7.2.2.2.2 searchEvent: EventFeed: Calls the search method from the SearchBar class
	1.2.1.7.2.2.1.2.1 search: SearchBar: Takes a list of events and a search string as parameters and returns a list of events that match the search query
	1.2.1.7.2.2.2.3 filterEvents: EventFeed: Calls the filter method from the Filter class
	1.2.1.7.2.2.1.3.1 filter: Filter: Takes a list of events as a parameter and return a list of events that match the filter options attribute
	1.2.1.7.2.2.2.4 displayEventFeed: EventFeed: Displays all the elements to the screen
	1.2.1.7.2.2.1.2.2 displaySearchBar: SearchBar
	1.2.1.7.2.2.2.5 displayEvents: EventFeed
	1.2.1.7.2.2.1.3.2 displayFilter: Filter
Referenced By:	1.2.1.7, 1.2.1.7.2
Viewpoint:	Structure Chart

View 1.2.1.7.2.2.3: Event List API Call

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "EventList",
  "type": "object",
  "properties": {
    "events": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "organizerID": { "type": "string" },
          "title": { "type": "string" },
          "description": { "type": "string" },
          "price": { "type": "number" },
          "icon": { "type": "string" },
          "addressID": { "type": "string" },
          "startTime": { "type": "string", "format": "date-time" },
          "endTime": { "type": "string", "format": "date-time" },
          "active": { "type": "boolean" }
        },
        "required": ["organizerID", "title", "description", "price", "icon", "addressID", "startTime",
"endTime", "active"]
      }
    }
  }
}
```

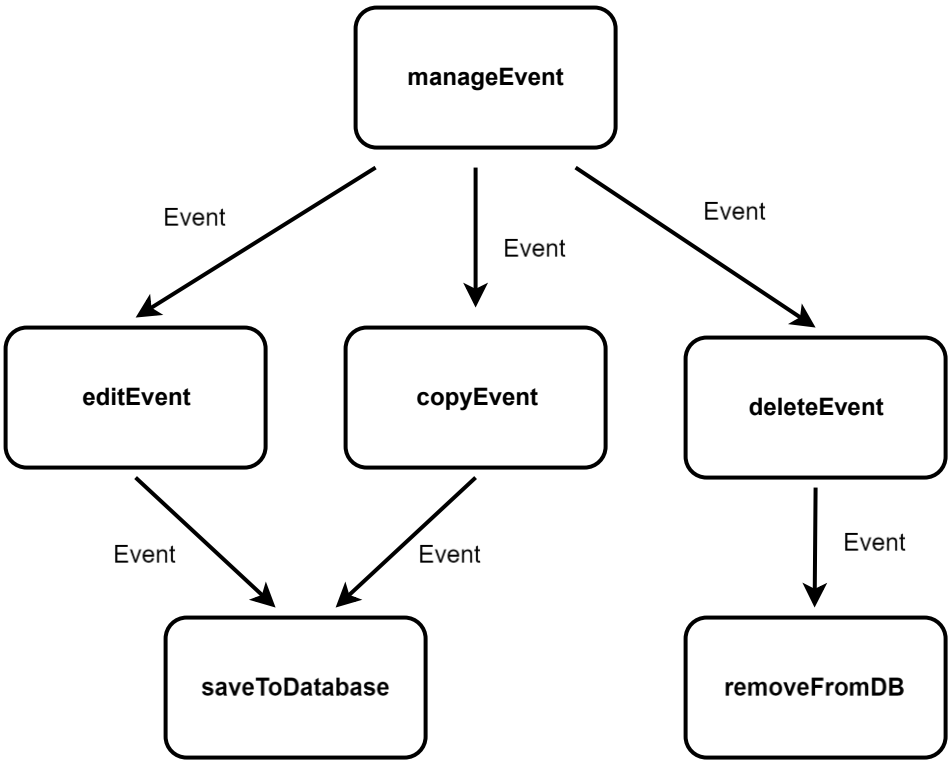
Name	**1.2.1.7.2.2.3 Event List API Call**
Purpose:	Provides a way to access event data from the database
Description:	An API call to get a list of events to display
Requirements:	PO.1.2.8, PO.1.2.9, PO.1.2.14
Elements:	**1.2.1.7.2.2.3 Events:** A list of events
	1.1.1.3 Event: An event stored in the database
Referenced By:	1.2.1.7.2.2.1
Viewpoint:	JSON Schema

View 1.2.1.7.3 Mobile Event Details



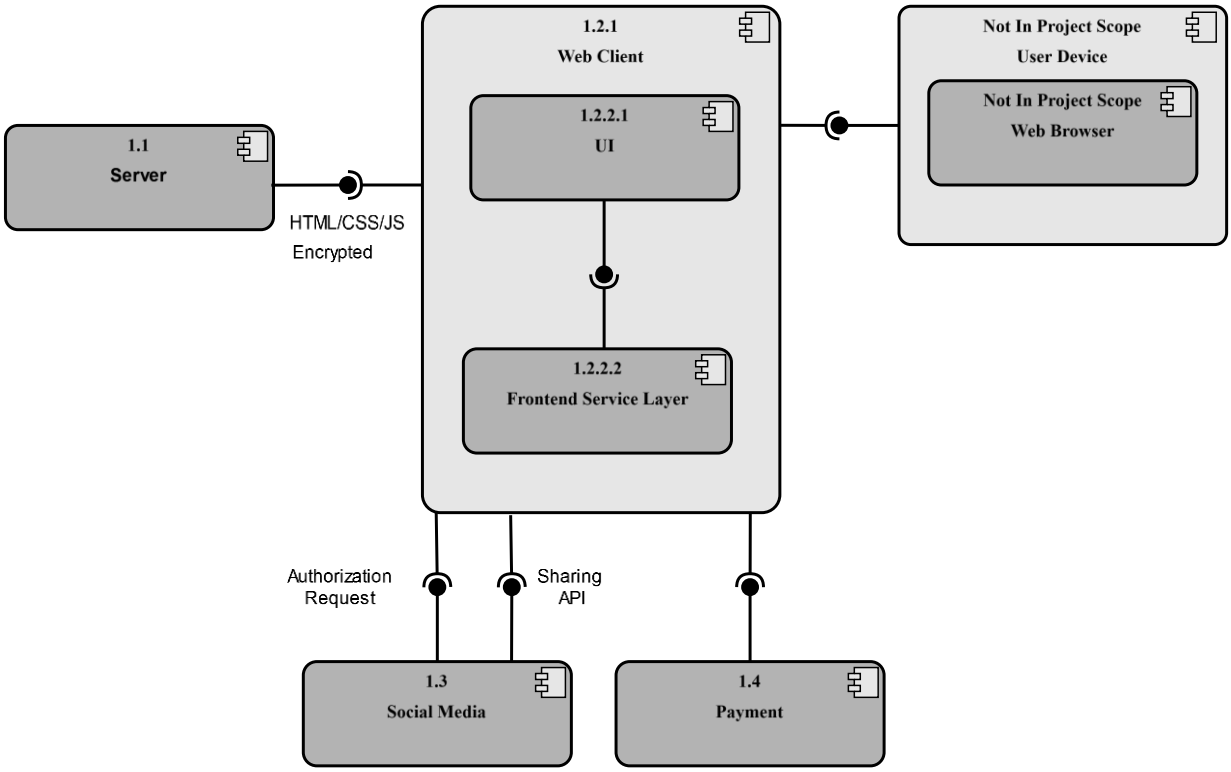
Name	**1.2.1.7.3 Mobile Event Details**
Purpose:	Provide the details about an individual event
Description:	The event details component contains all information pertaining to an event. It includes event-specific data such as the name, description, number of participants, duration, price, location, date, time, and an image. This component supports actions like updating and viewing event details.
Requirements:	PO.1.2.4, PO.1.2.7, PO.1.2.13, PO.1.2.14, PO.1.7.6
Elements:	**1.2.1.7.1.2 Set Event Info**
	1.2.1.7.3.1 Display
Referenced By:	1.2.1.7
Viewpoint:	Class Diagram

View 1.2.1.7.4 Mobile Event Management



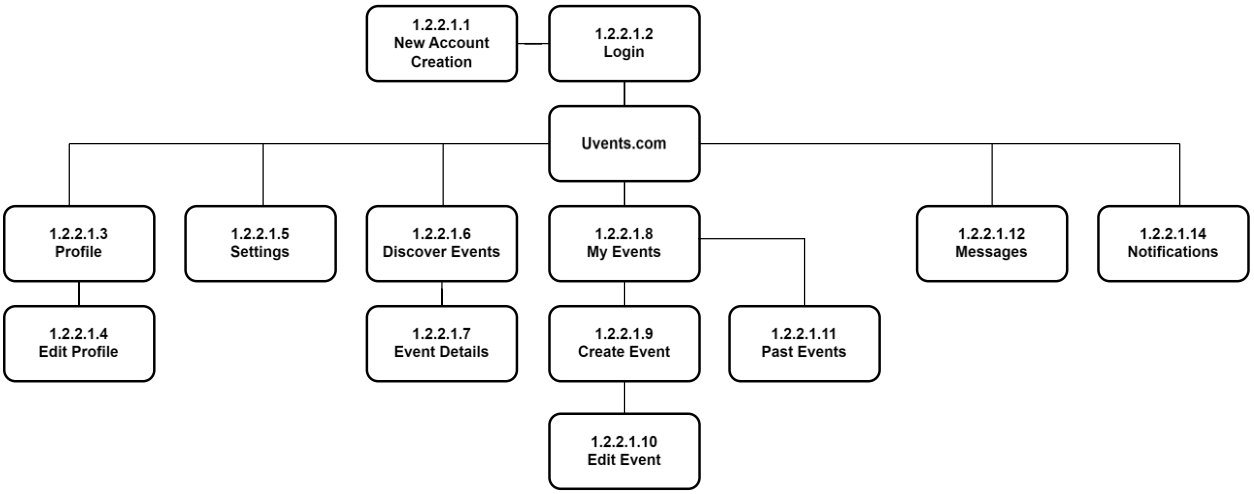
Name	1.2.1.7.4 Mobile Event Management
Purpose:	Show how a user's events can be managed
Description:	This component provides functionalities related to event management. It serves as an interface for users to interact with events, allowing them to create, edit, and delete events. These functionalities extend to commercial, consumer, and admin users.
Requirements:	PO.1.2.7, PO.1.7.9, PO.1.7.10, PO.1.7.11, PO.1.7.26, PO.1.7.35, PO.2.1.11, PO.2.2.12, PO.2.2.13, PO.2.3.8
Elements:	**1.2.1.7.4.1 editEvent**
	1.2.1.7.4.2 copyEvent
	1.2.1.7.4.3 saveToDatabase
	1.2.1.7.4.4 deleteEvent
	1.2.1.7.4.5 removeFromDB
Referenced By:	1.2.1.7
Viewpoint:	Structure Chart

View 1.2.2: Web Client



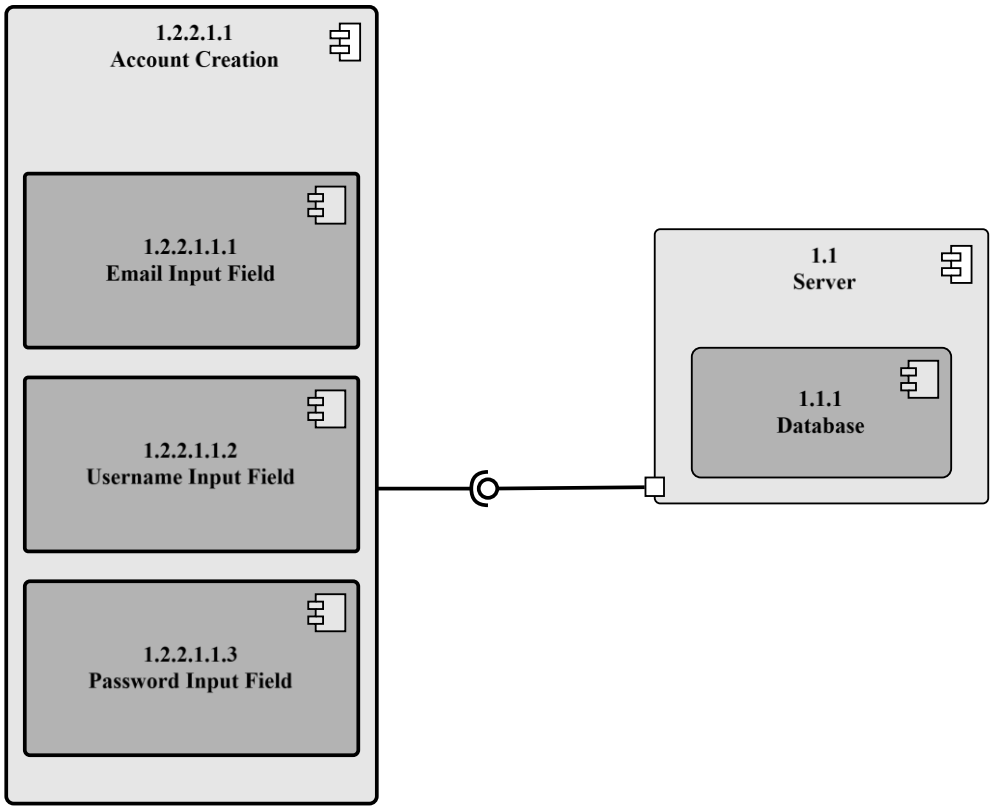
Name	1.2.2 Web Client Overview
Purpose:	Illustrate the components of the web client and how they interact with each other.
Description:	This view illustrates the main components of the web client and its sub-components.
Requirements:	New Requirement from the client, fulfills requirements for the Web Client.
Elements:	1.2.2.1 UI
	1.2.2.2 Frontend Service Layer
	Reference Only User Device and Web Browser
	1.1 Server
	1.3 Social Media
	1.4 Payment
Referenced By:	**1 Uvents Component Diagram**
Viewpoint:	Component Diagram

View 1.2.2.1: Web Client UI



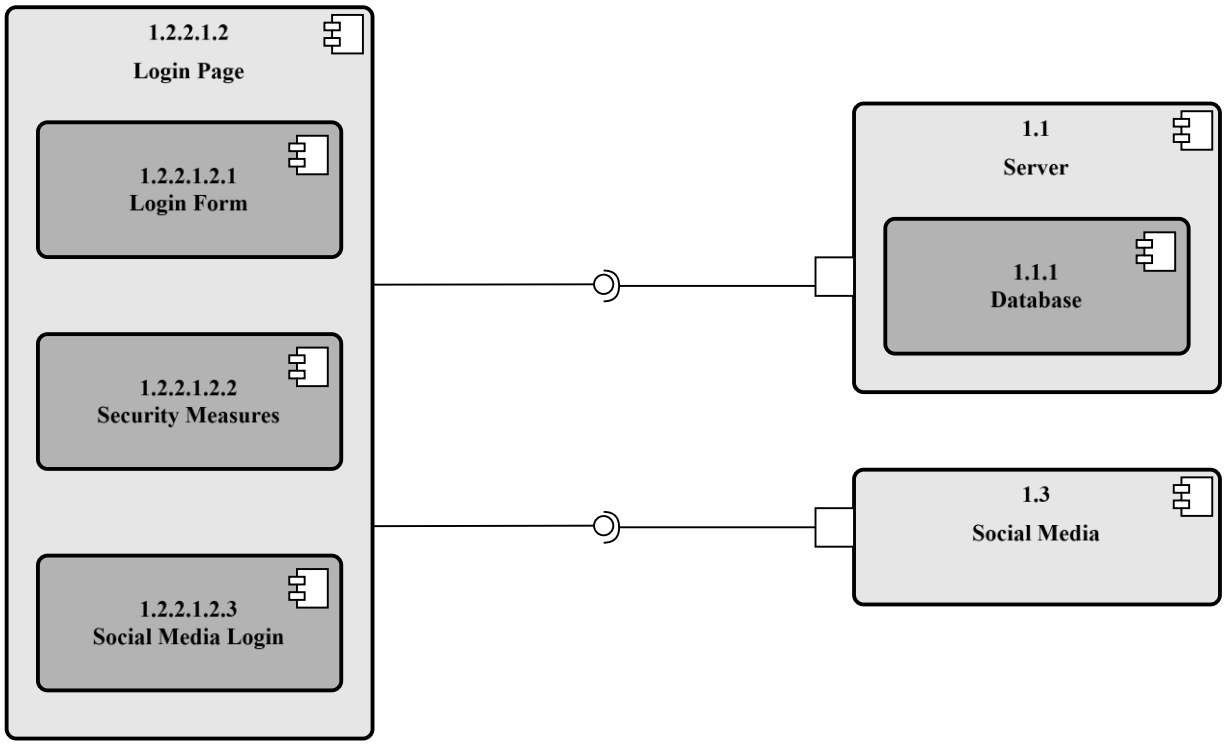
Name	**1.2.2.1 Web Client UI**	
Purpose	Describe the required Web Client UI in terms of the needed pages.	
Description	This is an overall look at the required pages that will be needed for the Web Client UI.	
Requirements	FS.1, FS.4 - FS.11, FS.13, FS.21, FS.23	
Elements	1.2.2.1.1 New Account Creation	1.2.2.1.8 My Events Page
	1.2.2.1.2 Login Page	1.2.2.1.9 Create Event Page
	1.2.2.1.3 Profile Page	1.2.2.1.10 Edit Event Page
	1.2.2.1.4 Edit Profile Page	**1.2.2.1.11 Past Events Page**
	1.2.2.1.5 Settings Page	1.2.2.1.12 Messages Page
	1.2.2.1.6 Discover Events Page	1.2.2.1.14 Notifications
	1.2.2.1.7 Event Details Page	**1.2.2.1.15 Auctions Page**
Referenced By	1.2.2 Web Client	
Viewpoint	Component Diagram	

View 1.2.2.1.1: Web Client Account Creation Page



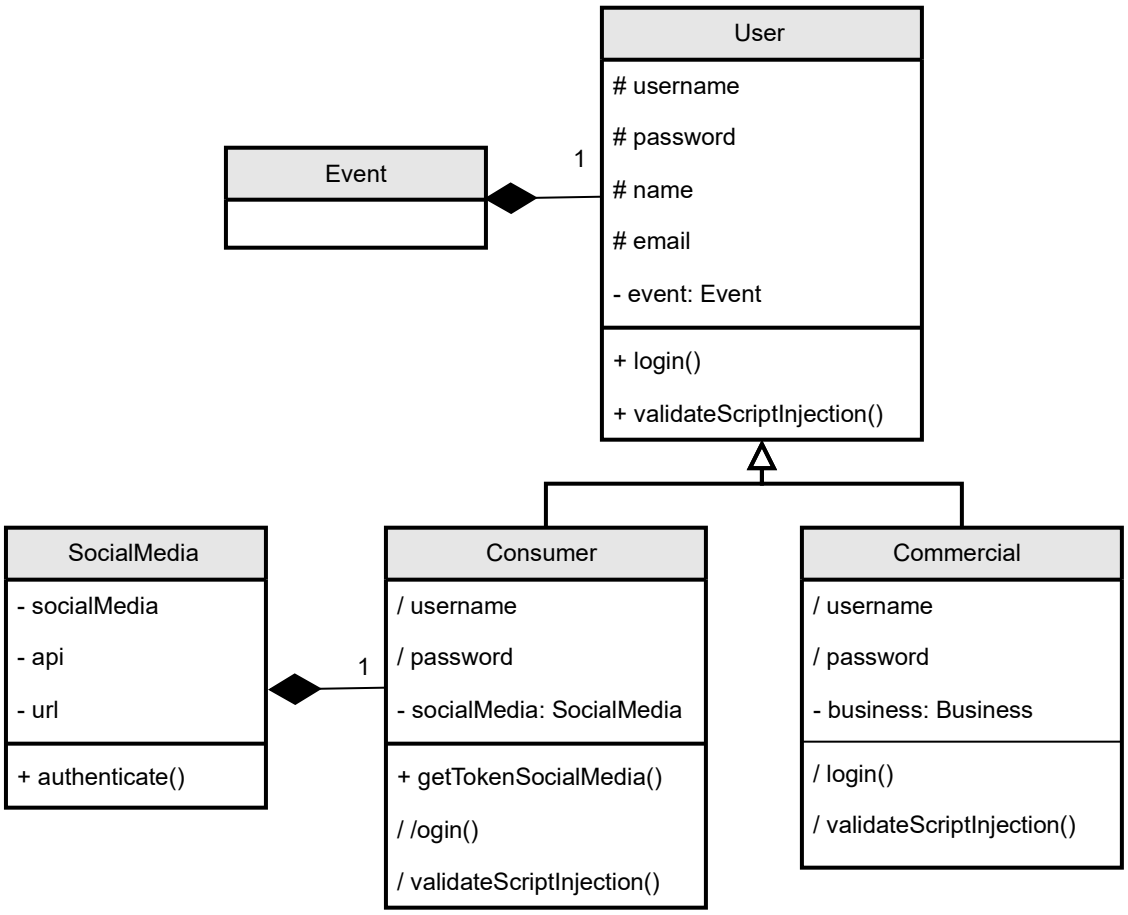
Name	1.2.2.1.1 Web Client Account Creation Page
Purpose:	Illustrates the component to the account creation page.
Description:	Illustrates the main components to the account creation page as well as how it interacts with other components.
Requirements:	FS.1 [PO.2.1.1], FS.15 [PO.2.2.1]
Elements:	**1.2.2.1.1.1 Email**
	1.2.2.1.1. 2 Username
	1.2.2.1.1. 3 Password
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.2: Web Client Login Page



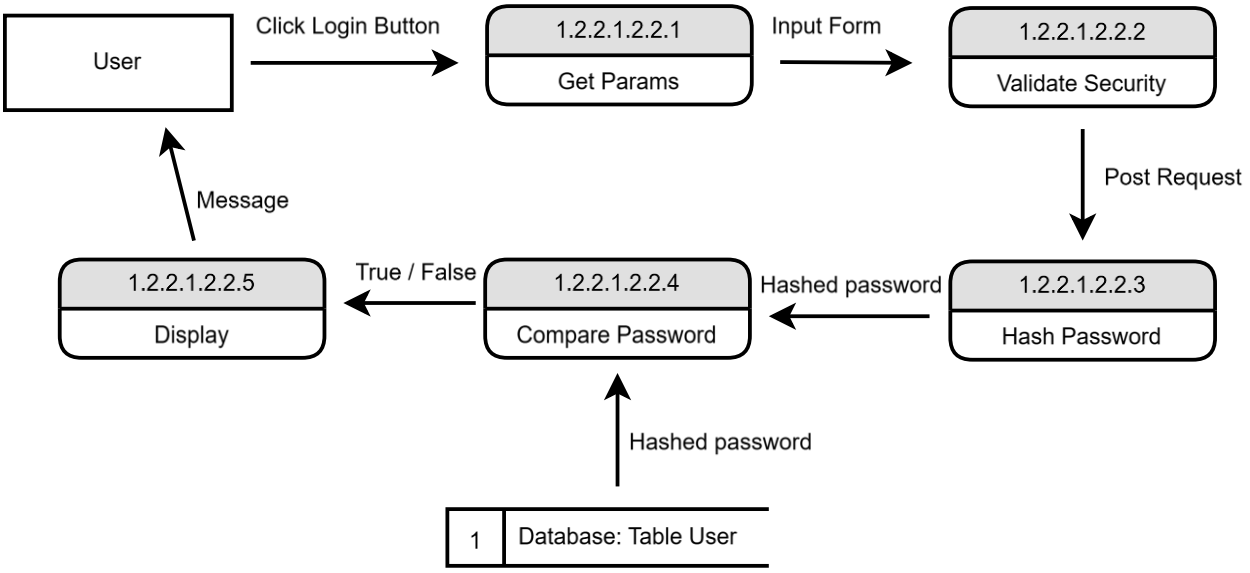
Name:	1.2.2.1.2: Web Client Login Page
Purpose:	Illustrates the components for the log in page
Description:	Illustrate all the main components to the log in page as well as how it interacts with other components
Requirements:	FS.6 [PO.2.1.6], FS.18 [PO.2.2.9], PO.1.2.1, PO.1.4.1, PO.1.7.1, PO.1.7.18, PO.2.1.6 [IR.6, IR.7, IR.13, IR.16], PO.2.2.9 [IR.5]
Elements:	1.2.2.1.2.1 **Login Form:** Form to allow user to login
	1.2.2.1.2.2 **Security Measures:** Validation to avoid XSS, SQL Injection and other security considerations
	1.2.2.1.2.3 Social Media Login: Connection to various social media
Referenced By:	1.2.2.1 UI
Viewpoint:	Component Diagram

Viewpoint 1.2.2.1.2.1: Class Diagram for Login



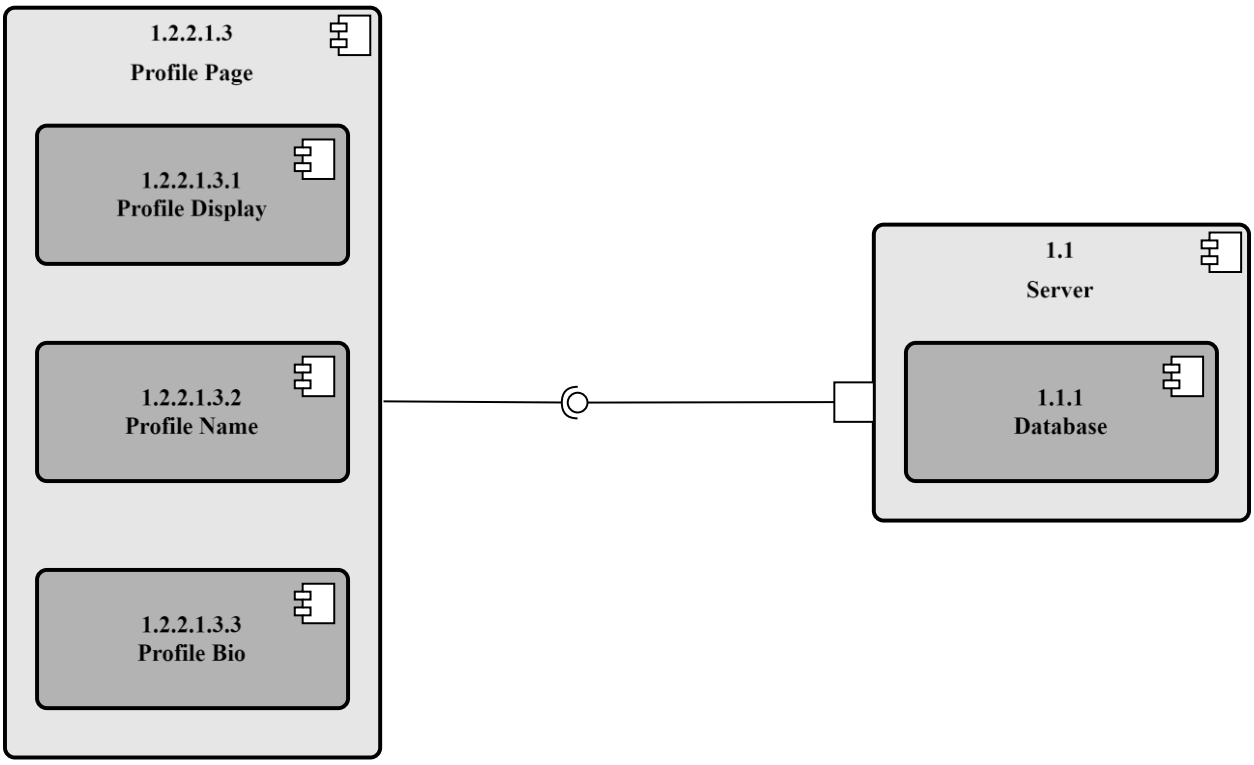
Name	**1.2.2.1.2.1 Class Diagram for Login**
Purpose:	Illustrate related classes for login
Description:	Login requires a class diagram to understand the objects that interact with authentication
Requirements:	FS.6 Consumer Login [PO.2.1.6], FS.18 Commercial Login [PO.2.2.9]
Elements:	**1.2.2.1.2.1.1 User:** Class for the user object
	1.2.2.1.2.1.2 Social Media: Class for the social media object
	1.2.2.1.2.1.3 Consumer: Class inherited from user class for the consumer role
	1.2.2.1.2.1.4 Commercial: Class inherited from user class for the commercial role
	1.2.2.1.2.1.5 Event: List of user events
Referenced By:	1.2.2.1.2 View Component Login
Viewpoint:	Class Diagram

Viewpoint 1.2.2.1.2.2: Data Flow Diagram for Login



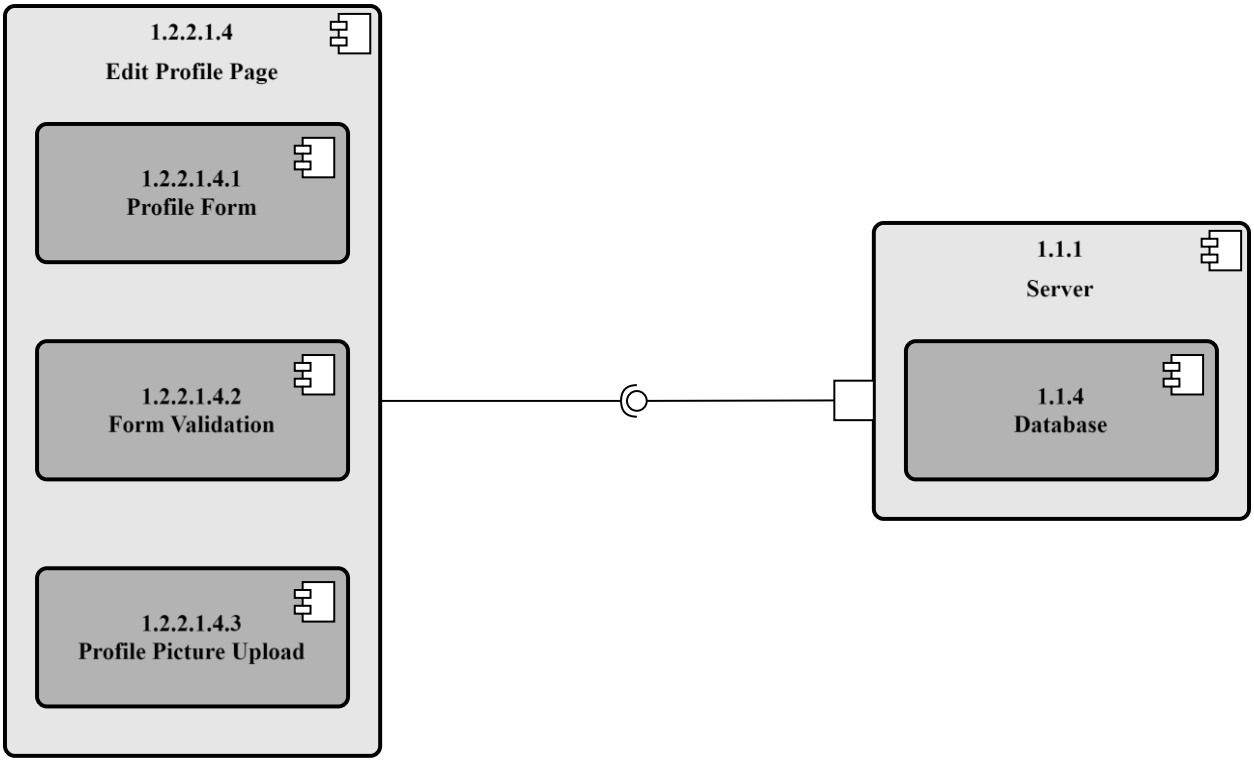
Name	1.2.2.1.2.2 Data Flow Diagram for Login
Purpose:	Illustrate data flow process for login
Description:	Login requires a diagram showing the data flow in authentication.
Requirements:	FS.6 Consumer Login [PO.2.1.6], FS.18 Commercial Login [PO.2.2.9]
Elements:	1.2.2.1.2.2.1 **Get Params:** Get the user and password. **1.2.2.1.2.2.2 Validate Security:** Validate the input to check script injections like XSS and SQL Injection. **1.2.2.1.2.2.3 Hash Password** **1.2.2.1.2.2.4 Compare Password** **1.2.2.1.2.2.5 Display:** Display a message to the user.
Referenced By:	1.2.2.1.2 View Component Login
Viewpoint:	Data Flow Diagram

View 1.2.2.1.3: Web Client Profile Page



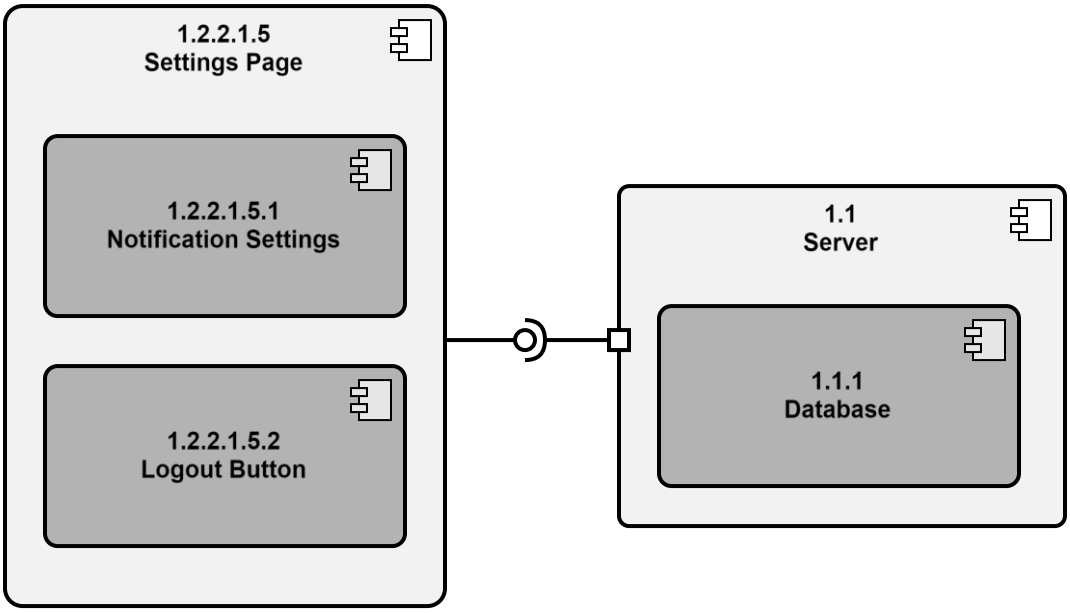
Name:	**1.2.2.1.3 Web Client Profile Page**
Purpose:	Illustrates the component to the profile page.
Description:	This view shows the main components of the profile page and how it interacts with other components.
Requirements:	FS.5 View My Profile [PO.2.1.5], FS.17 View My Profile [PO.2.2.8]
Elements:	**1.2.2.1.3.1 Profile Display**
	1.2.2.1.3.2 Profile Name
	1**.2.2.1.3.3 Profile Bio**
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.4 Web Client Edit Profile Page



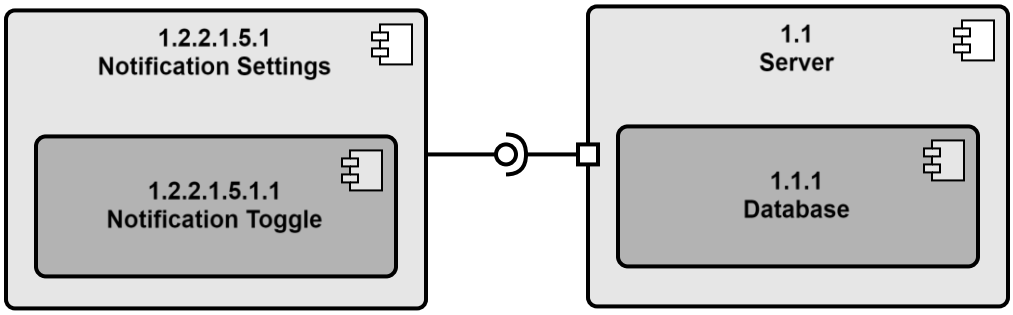
Name:	**1.2.2.1.4: Web Client Edit Profile Page**
Purpose:	Illustrates the components of the web client edit profile page.
Description:	This view illustrates the main components to the Edit Profile Page and their subcomponents.
Requirements:	PO.1.2.6, PO.1.7.3, PO.1.7.24, PO.2.1.4, PO.2.2.5, PO.2.2.8, PO.2.3.3, PO.3.4.2, PO.3.5.2, FS.4, FS.4.1, FS.4.2, FS.4.3, FS.4.4, FS.4.5, FS.16, FS.28, DSC.1.2, FV.4, FV.28, DCV.1.2
Elements:	**1.2.2.1.4.1 Profile Form**
	1.2.2.1.4.2 Form Validation
	1.2.2.1.4.3 Profile Picture Upload
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.5 Web Client Settings Page



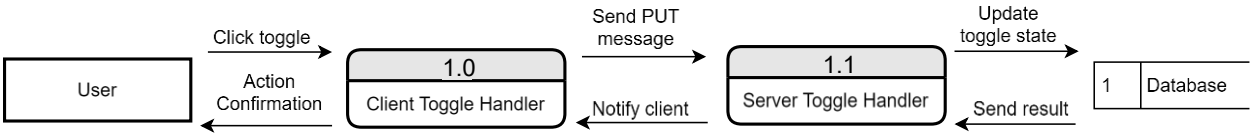
Name:	1.2.2.1.5 Web Client Settings Page
Purpose:	Represent the components for the Web Client Settings page.
Description:	Represent the main components for the settings page and illustrate how they interact with other components.
Requirements:	FS.13 [PO.2.1.15], FS.24 [PO.2.2.19]
Elements	1.2.2.1.5.1 Notification Settings
	1.2.2.1.5.2 Logout Button
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.5.1 Web Client Notification Settings



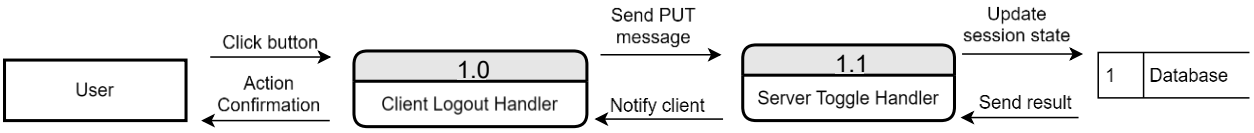
Name:	1.2.2.1.5.1 Web Client Notification Settings
Purpose:	Represent the components for the web client notification settings.
Description:	Represent the main components for the notification settings, which consists of two toggles that users can opt out of certain notification types.
Requirements:	FS.13.1
Elements	1.2.2.1.5.1.1 Notification Toggle
Referenced By:	1.2.2.1.5 Web Client Settings Page
Viewpoint:	Component Diagram

View 1.2.2.1.5.1.1 Web Client Notification Toggle



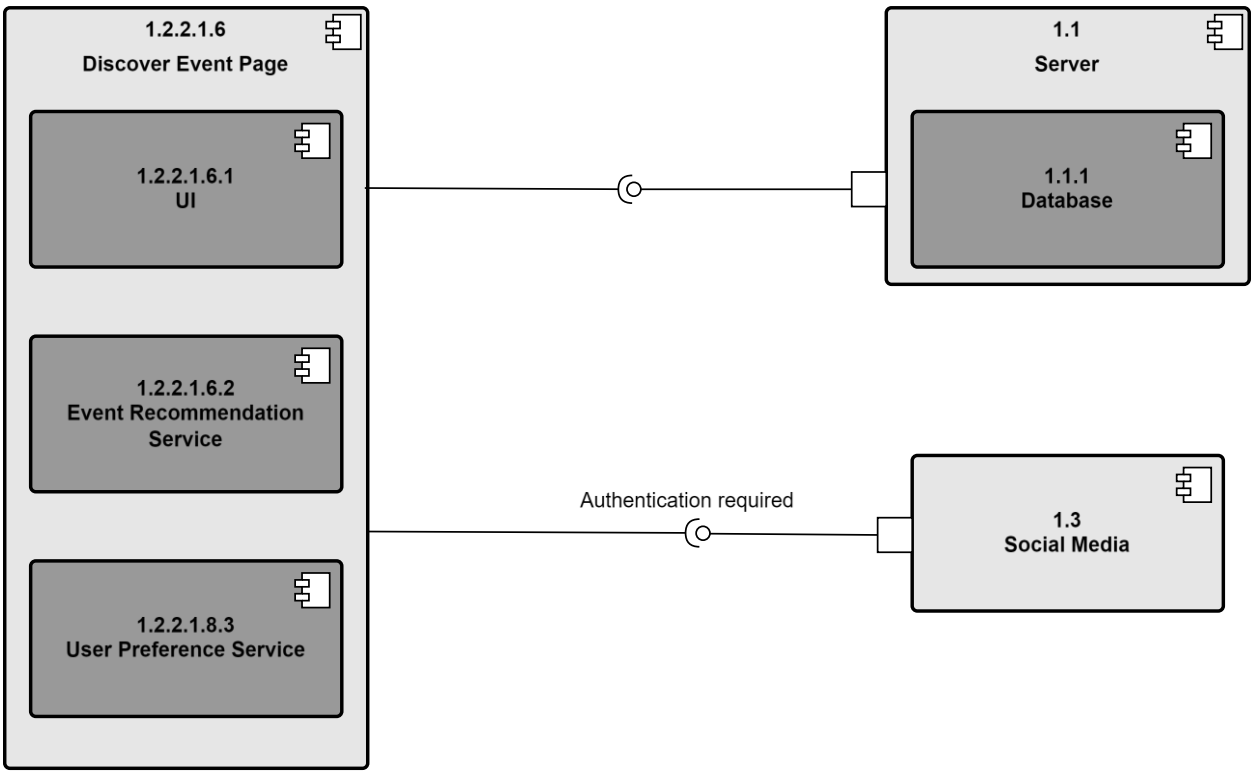
Name:	**1.2.2.1.5.1.1 Web Client Notification Toggle**
Purpose:	Represent the flow of data when a user toggles a notification setting.
Description:	There are two types of notifications that the user can opt in/out of (Event Invites and Event Acceptance), but the flow of data will be similar for both.
Requirements:	FS.13.1
Elements	**1.0 Client Toggle Handler, 1.1 Server Toggle Handler**
Referenced By:	1.2.2.1.5.1 Web Client Notification Settings
Viewpoint:	Data Flow Diagram

View 1.2.2.1.5.2 Web Client Settings Page Logout



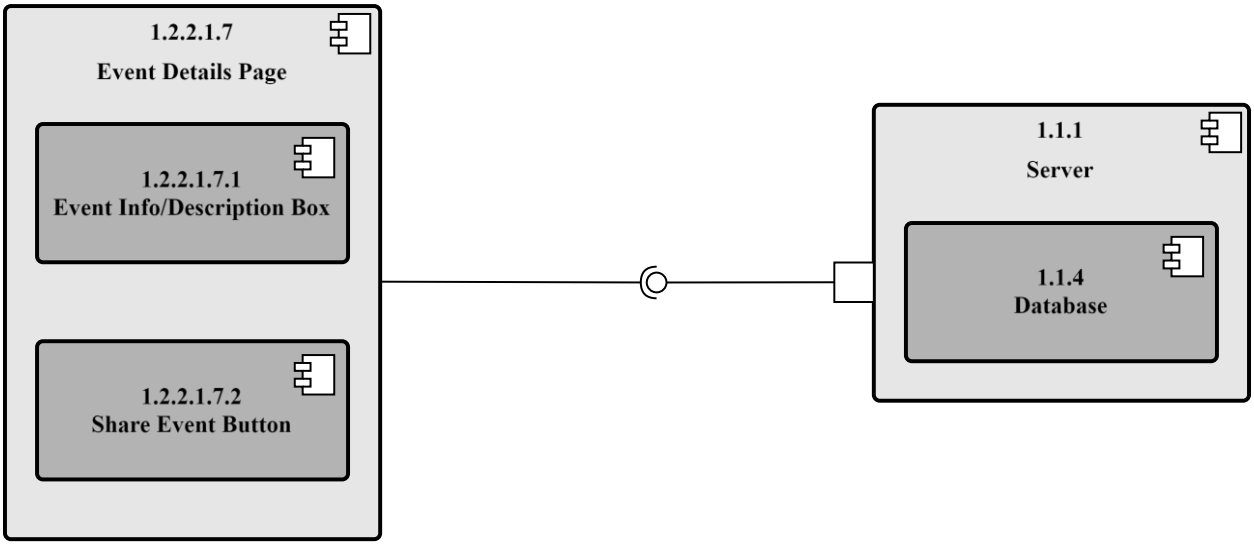
Name:	**1.2.2.1.5.2 Web Client Settings Page Logout**
Purpose:	Represent the flow of data when a user clicks logout from the Settings page.
Description:	There are multiple places a user can log out from, and this represents the flow of data when the user is on the Settings page and logs out.
Requirements:	FS.13.2. FS.14.1, FS.14.2
Elements	**1.0 Client Logout Handler, 1.1 Server Logout Handler**
Referenced By:	**1.2.2.1.5 Web Client Settings Page**
Viewpoint:	Data Flow Diagram

Viewpoint 1.2.2.1.6 Discover Events Page



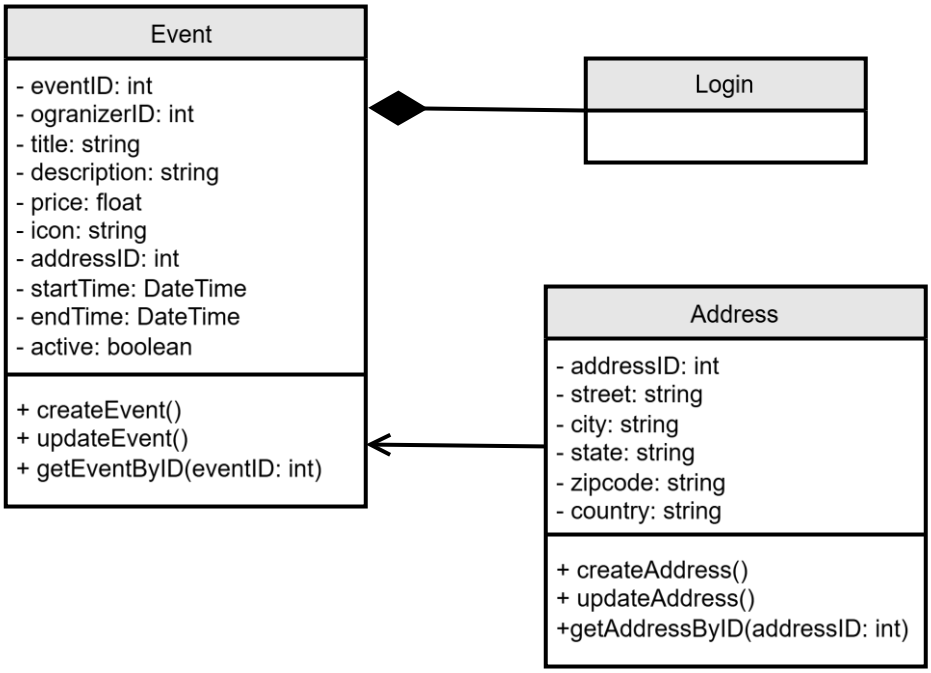
Name:	1.2.2.1.6: Web Client Discover Events Page
Purpose:	Illustrates the various components that make up the discover events page and how they interact.
Description:	This view illustrates the main components of the discover events page, as well as how they interact with other components.
Requirements:	FS.7 Discover [PO.2.1.7]
Elements:	**1.2.2.1.6.1 UI**
	1.2.2.1.6.2 Event Recommendation
	1.2.2.1.6.3 User Preference Service
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.7 Web Client Event Details Page



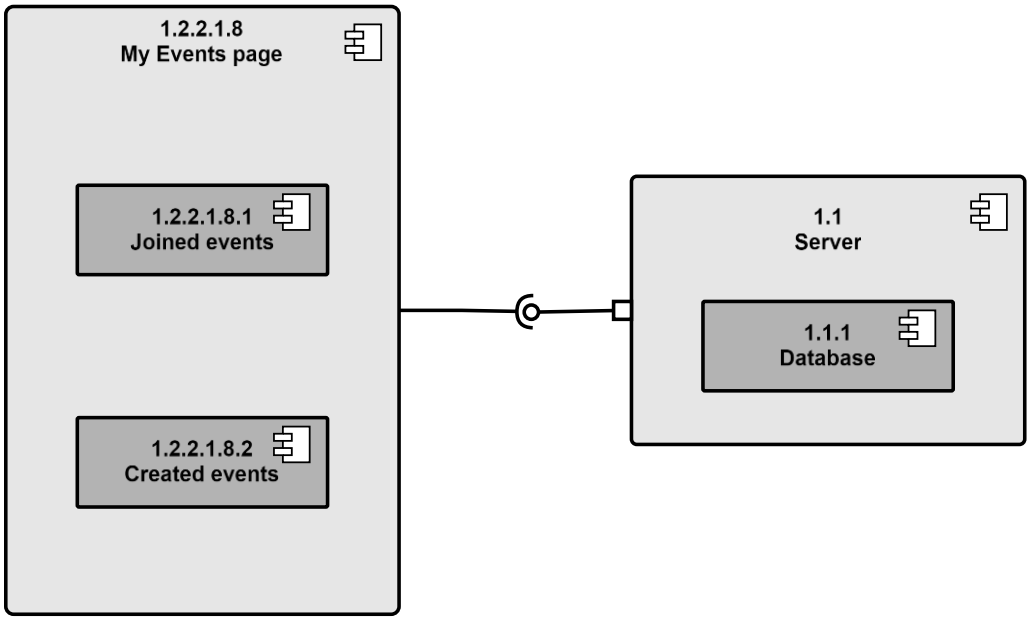
Name:	1.2.2.1.7: Web Client Event Details Page
Purpose:	Illustrate the components of the web client's Event Details page and how they interact with each other.
Description:	This view illustrates the main components to the web client's details page and their subcomponents.
Requirements:	FS.19.10.1, EIV.3.3, PO.1.2.13, PO.1.7.6
Elements:	**1.2.2.1.7.1 Event Info/Description Box**
	1.2.2.1.7.2 Share Event Button
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.2.7.1: Web Client Event Class Diagram



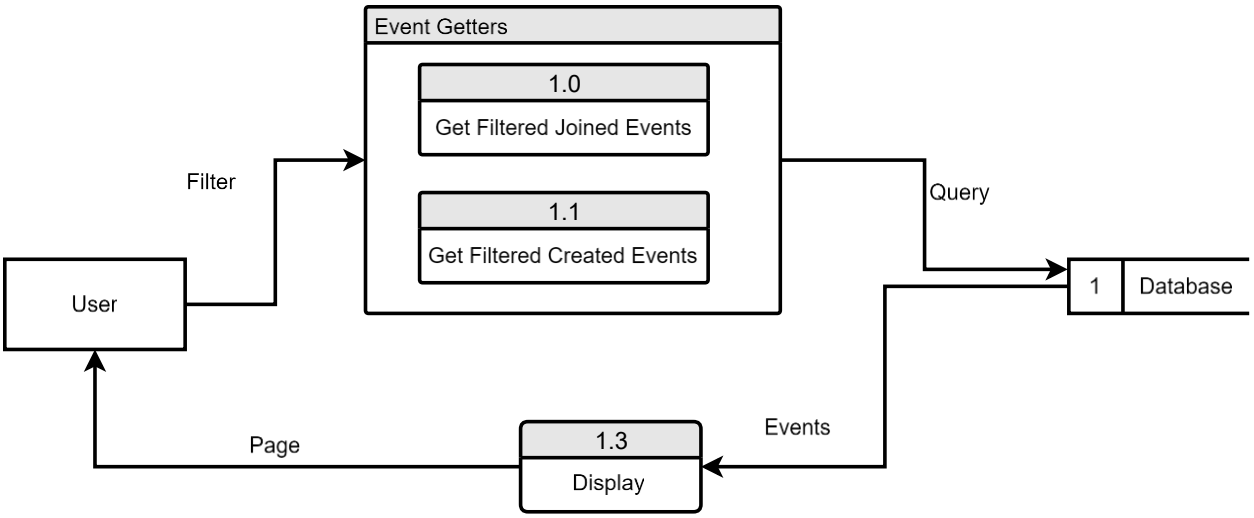
Name:	**1.2.2.1.2.7.1 Web Client Event Class Diagram**
Purpose:	To diagram the components of the Event class and how it relates to other classes.
Description:	The event class will contain many different attributes, be related to other classes through means such as inheritance and composition and will have methods.
Requirements:	LDR.3, LDV.3
Elements:	**1.2.2.1.2.7.1.1 Event: C**lass manages details for events, including attributes like event ID, organizer ID, title, description, price, icon, address ID, start and end times, and active status. It provides methods to create, update, and retrieve event information, linking events to Commercial User organizers and specific Address locations.
	1.2.2.1.2.7.1.2 Address: Class handles location details for events, storing attributes such as address ID, street, city, state, zip code, and country. It includes methods to create, update, and retrieve addresses, ensuring each event is associated with a precise location.
Referenced By:	1.2.2.1.7 Web Client Event
Viewpoint:	Class Diagram

View 1.2.2.1.8: Web Client My Events Page



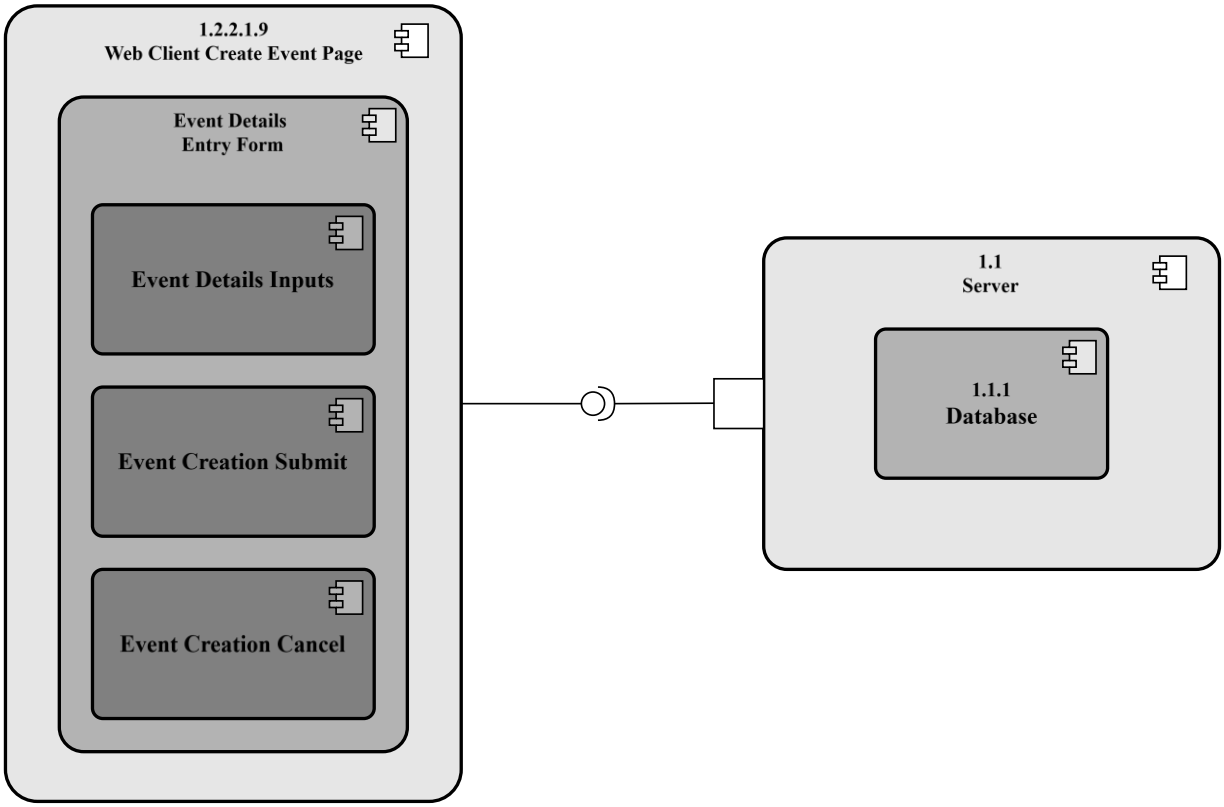
Name	View 1.2.2.1.8: Web Client My Events Page
Purpose:	Illustrates the components of the My Events page.
Description:	Illustrates the main components of the My Events page and how it interacts with others.
Requirements:	FS.11 [PO.2.1.13]
Elements:	**1.2.2.1.8.1 Joined Events:** A list of events the user has joined
	1.2.2.1.8.2 Created Events: A list of events the user has created
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.8.1: Web Client My Events Page Data Flow Diagram



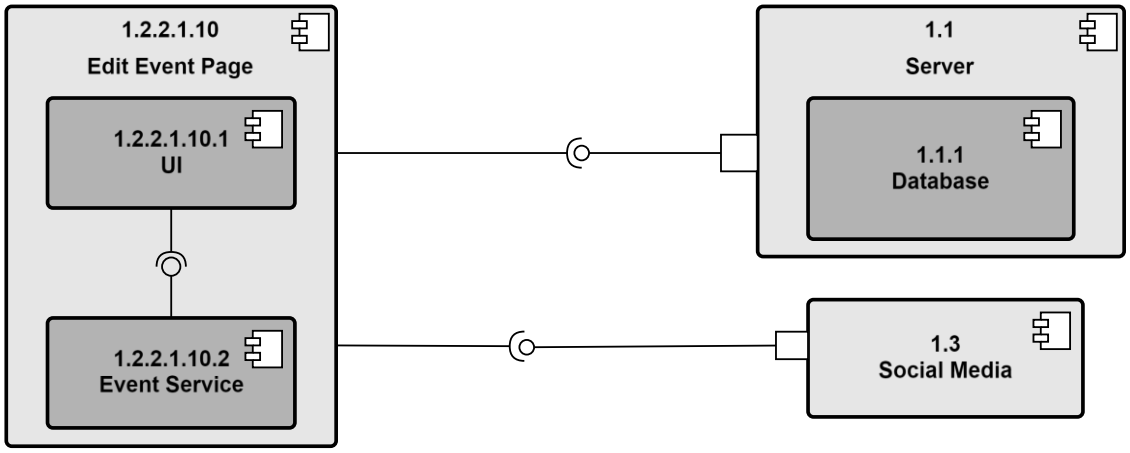
Name	View 1.2.2.1.8.1: Web Client My Events Page Data Flow Diagram
Purpose:	Illustrates the flow of data for the My Events page.
Description:	Illustrates the main flow of data between the user and database for the My Events page.
Requirements:	FS.11 [PO.2.1.13]
Elements:	**1.0 Get Filtered Joined Events:** Sends a query to the database to get a filtered list of joined events.
	1.2.2.1.8.2 Created Events: Sends a query to the database to get a filtered list of created events.
Referenced By:	1.2.2.1.8 Web Client My Events Page
Viewpoint:	Data flow Diagram

View 1.2.2.1.9: Web Client Create Event Page



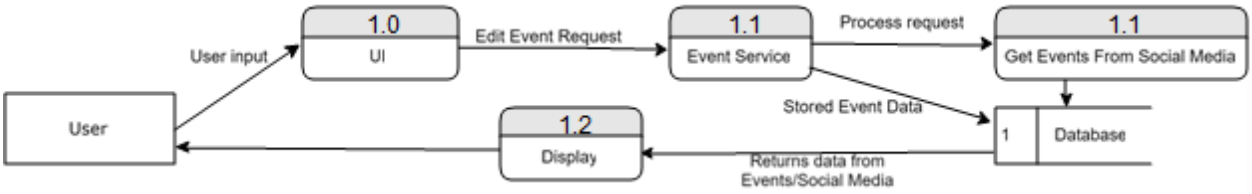
Name:	1.2.2.1.9 Web Client Create Event Page
Purpose:	This document's purpose is to record the structure and interactions of the Web Client Create Event Page which enables users to create events through a web client.
Description:	The Web Client Create Event Page
Requirements:	FS.8 [PO.2.1.10], FS.19 [PO.2.2.10]
Elements:	**Event Details Entry Form**
	Event Details Inputs
	Event Creation Submit
	Event Creation Cancel
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component diagram

View 1.2.2.1.10: Edit Event Page



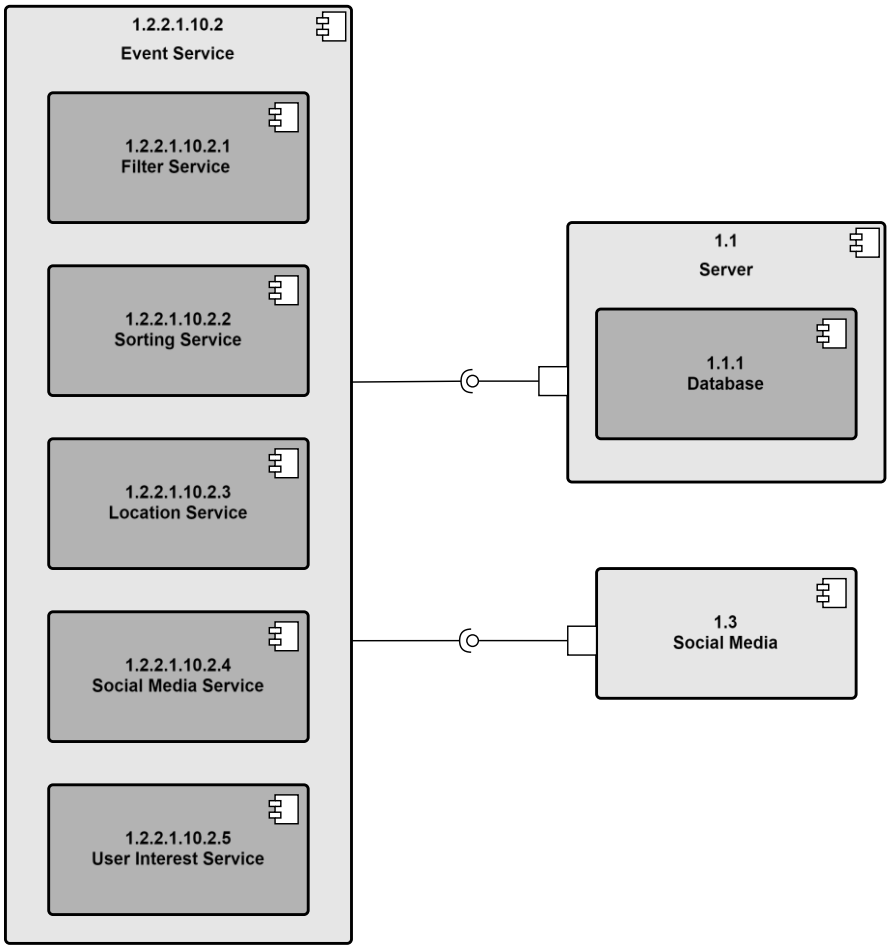
Name:	**1.2.2.1.10 Edit Event Page**
Purpose:	Illustrates the various components that make up the edit events page and how they interact.
Description:	This view illustrates the main components of the edit events page, as well as how they interact with other components.
Requirements:	FS.20
Elements:	1.2.2.1.10 Edit Event Page
	1.2.2.1.10.1 UI: Displays a list of events from Uvents and social media, allows users to filter and sort events, and shows detailed information about a selected event
	1.2.2.1.10.2 Event Service: Fetches data from database and interacts with the UI
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.10.1: Edit Event Diagram



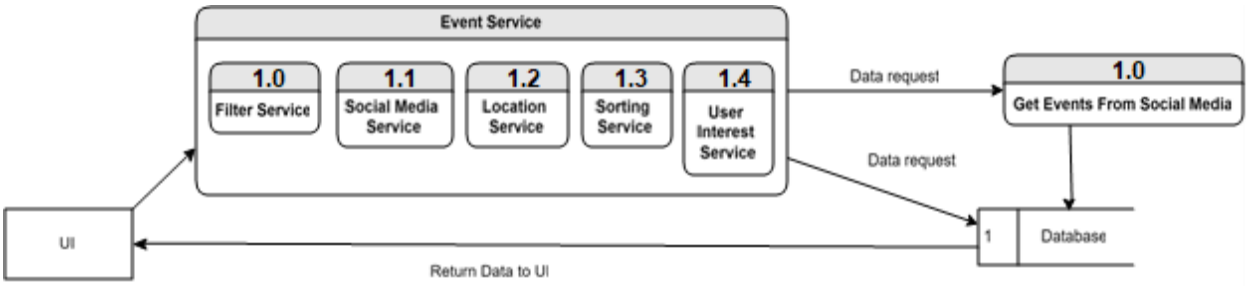
Name:	1.2.2.1.10.1 Edit Event Diagram
Purpose:	Illustrates the DFD components of the Edit Event Page.
Description:	This view illustrates the DFD for the Edit Event Page.
Requirements:	FS.7.2 Filtering/Sorting [PO.2.1.8]
Elements:	**1.0 UI**
	1.1 Event Service
	1.2 Display
Referenced By:	1.2.2.1.10 Edit Event Page
Viewpoint:	Data Flow Diagram

View 1.2.2.1.10.2: Event Service



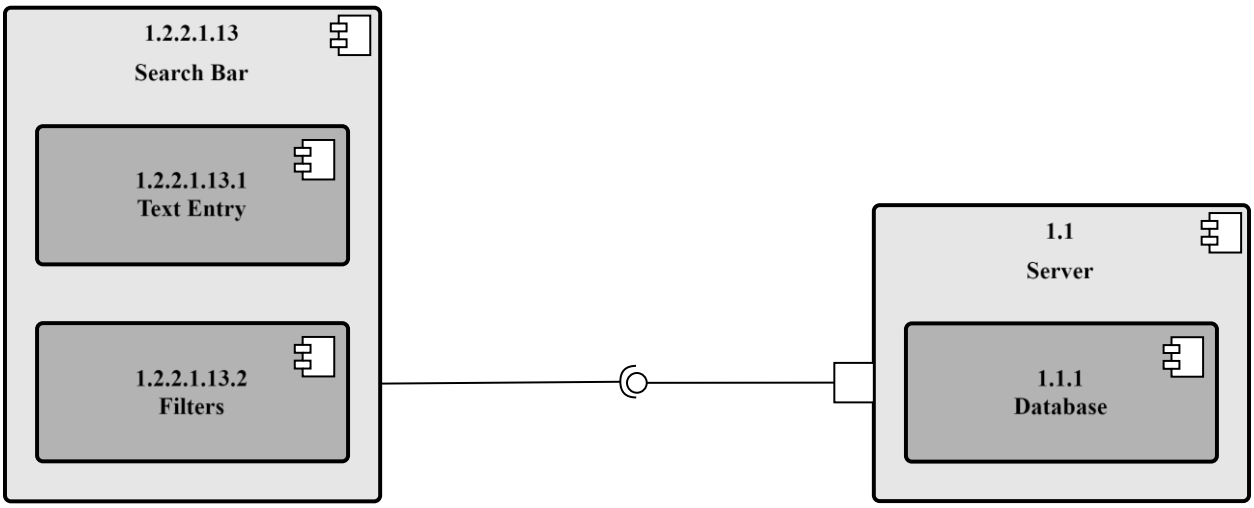
Name:	**1.2.2.1.10.2 Event Service**
Purpose:	Illustrates the various components that make up the Event Service and how they interact.
Description:	This view illustrates the main components of the Event Service, as well as how they interact with other components.
Requirements:	FS.7.2 Filtering/Sorting [PO.2.1.8]
Elements:	1.2.2.1.10.2.1 **Filter Service:** Applies filters (type, location) to events.
	1.2.2.1.10.2.2 Sorting Service: Sorts events based on criteria such as date, popularity.
	1.2.2.1.10.2.3 Location Service: Filters events based on location criteria.
	1.2.2.1.10.2.4 Social Media Service: Integrates with social media APIs to fetch friends' event data and post shares.
	1.2.2.1.10.2.5 User Interest Service: Determines user interests to show relevant events.
Referenced By:	1.2.2.1.10 Edit Event Page
Viewpoint:	Component Diagram

View 1.2.2.1.10.2.1: Event Service



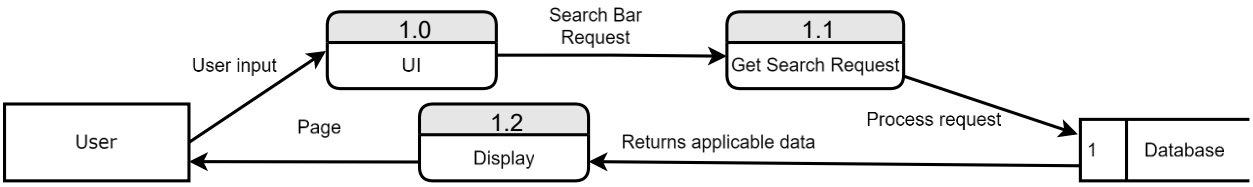
Name:	1.2.2.1.10.2.1 Event Service
Purpose:	Illustrates the DFD components of the Event Service.
Description:	This view illustrates the DFD for the Event Service.
Requirements:	FS.7.2 Filtering/Sorting [PO.2.1.8]
Elements:	**1.0 Filter Service**
	1.1 Social Media
	1.2 Location Service
	1.3 Sorting Service
	1.4 User Interest Service
Referenced By:	1.2.2.1.10 Edit Event Page
Viewpoint:	Data Flow Diagram

View 1.2.2.1.13 Web Client Search Bar



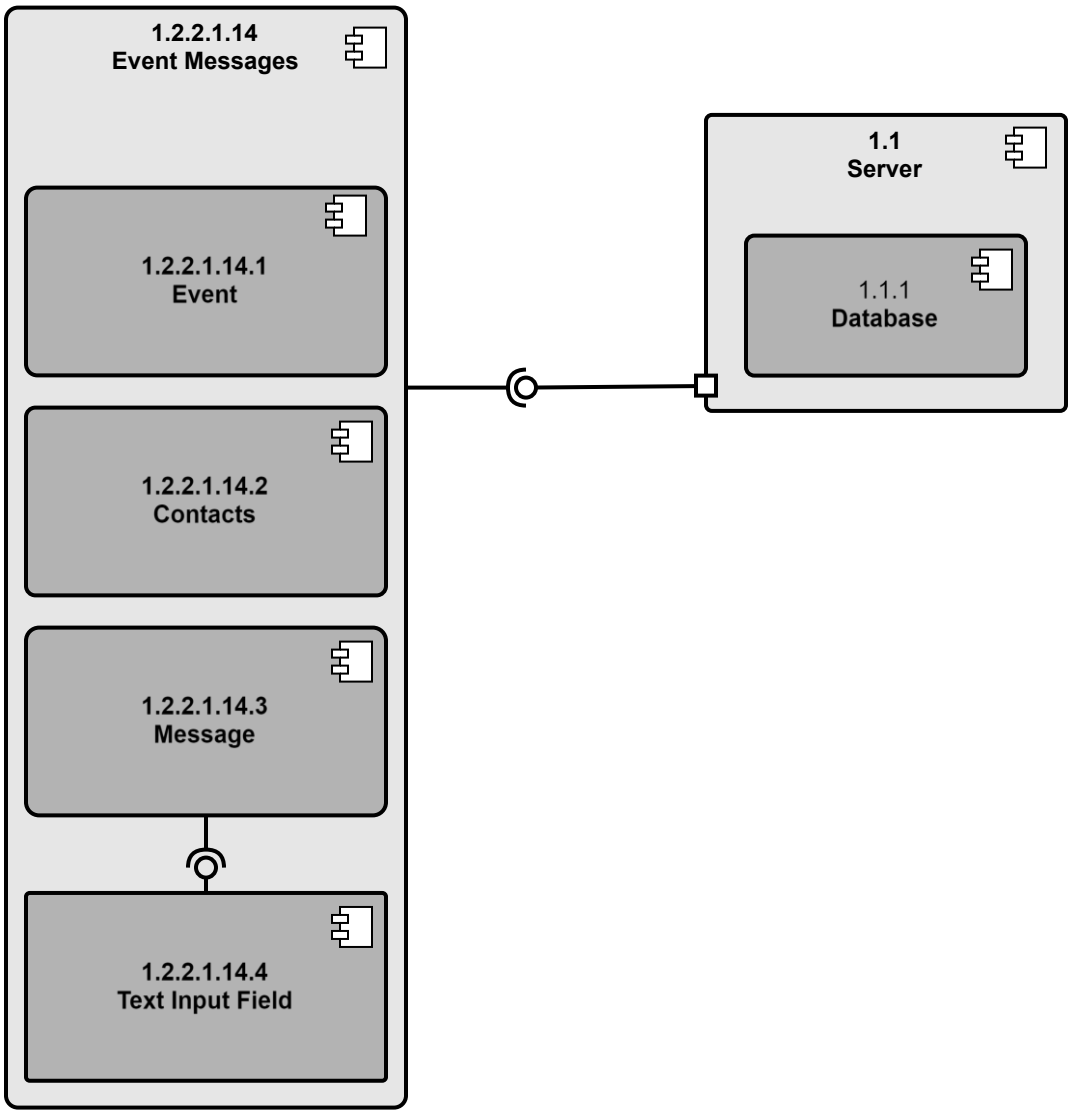
Name:	**1.2.2.1.13: Web Client Search Bar**
Purpose:	Illustrate the components of the web client search bar and how they interact with each other.
Description:	This view illustrates the main components of the web client search bar and their sub components.
Requirements:	PO.1.2.8, PO.1.7.28, PO.2.1.9, FS.7.2, FS.7.3, FV.7.3
Elements:	**1.2.2.1.13.1 Text Entry**
	1.2.2.1.13.2 Filters
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

View 1.2.2.1.13.1 Web Client Search Bar DFD



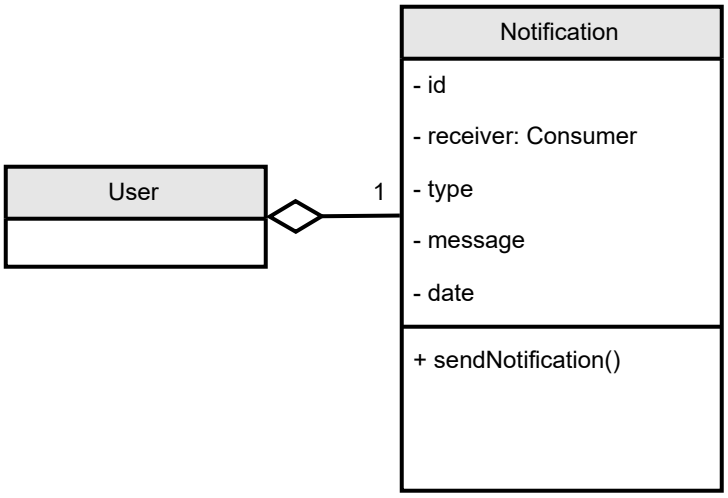
Name:	**1.2.2.1.13.1: Web Client Event Search Bar DFD**
Purpose:	Illustrate the data flow of the web client search bar and how it moves data with other parts of the program.
Description:	This view illustrates the main components of the web client search bar and its dataflow.
Requirements:	PO.1.2.8, PO.1.7.28, PO.2.1.9, FS.7.2, FS.7.3, FV.7.3
Elements:	**1.0 UI:** Gathers input and waits for event to start process then information to create a get request.
	1.1 Get Search Request: A search request is sent to the server with the applicable information.
Referenced By:	1.2.2.1.13 Web Client Search Bar
Viewpoint:	Data flow Diagram

View 1.2.2.1.14: Web Client Event Messages



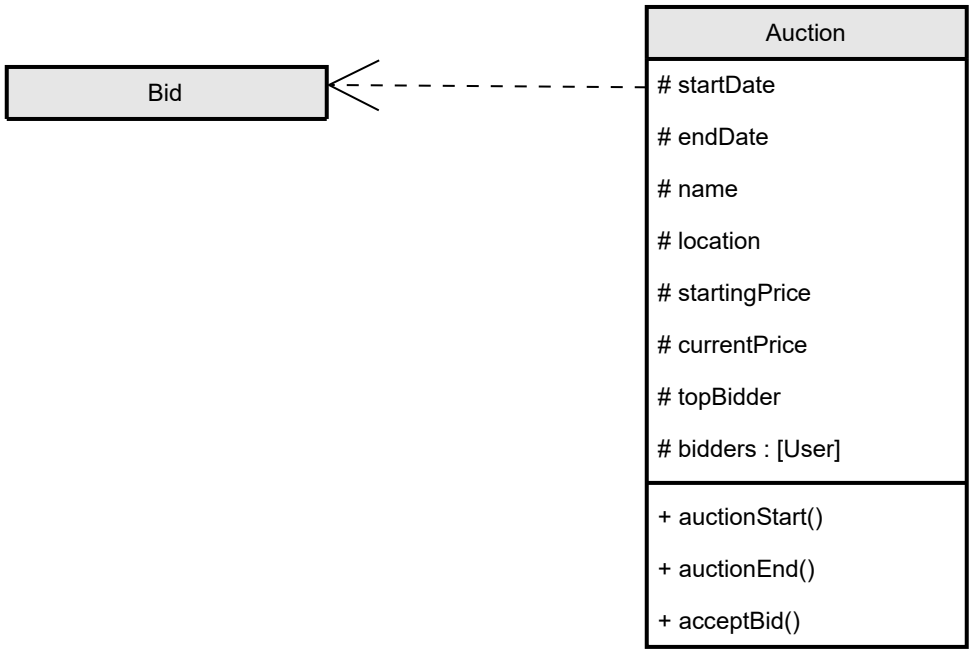
Name:	1.2.2.1.14: Event Messages
Purpose:	Illustrate the components to the messages page.
Description:	Illustrates the main components to the event messages and how they interact with other components.
Requirements:	FS.10 [PO.2.1.12]
Elements:	1.2.2.1.14.1 **Event**
	1.2.2.1.14.2 Contacts
	1.2.2.1.14.3 Messages
	1.2.2.1.14.4 Text input
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Component Diagram

Viewpoint 1.2.2.1.14.1: Notification Class Diagram



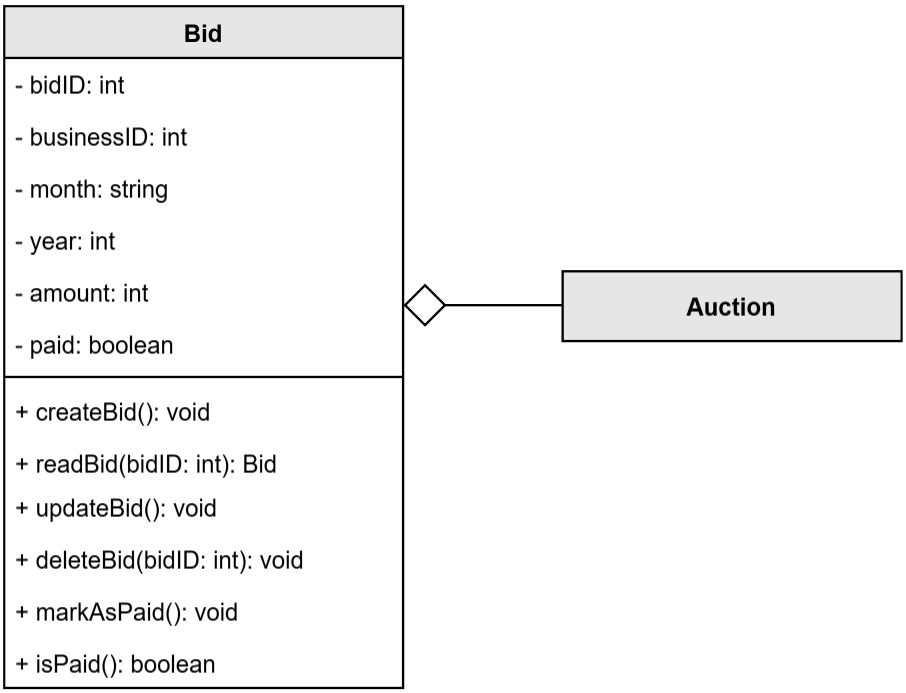
Name	1.2.2.1.14.1 Notification Class Diagram
Purpose:	To diagram the components of the notification class and how it relates to other classes
Description:	The notification class will contain many different attributes, be related to other classes through means such as inheritance and composition and will have methods.
Requirements:	LDR.11 Notification Table / FS.36 Notifications [PO.2.3.11]
Elements:	**1.2.2.1.14.1.1 Notification**
	1.2.2.1.14.1.2 User
Referenced By:	1.2.2.1.2 View Component Login
Viewpoint:	Class Diagram

Viewpoint 1.2.2.1.15.1: Auction Class Diagram



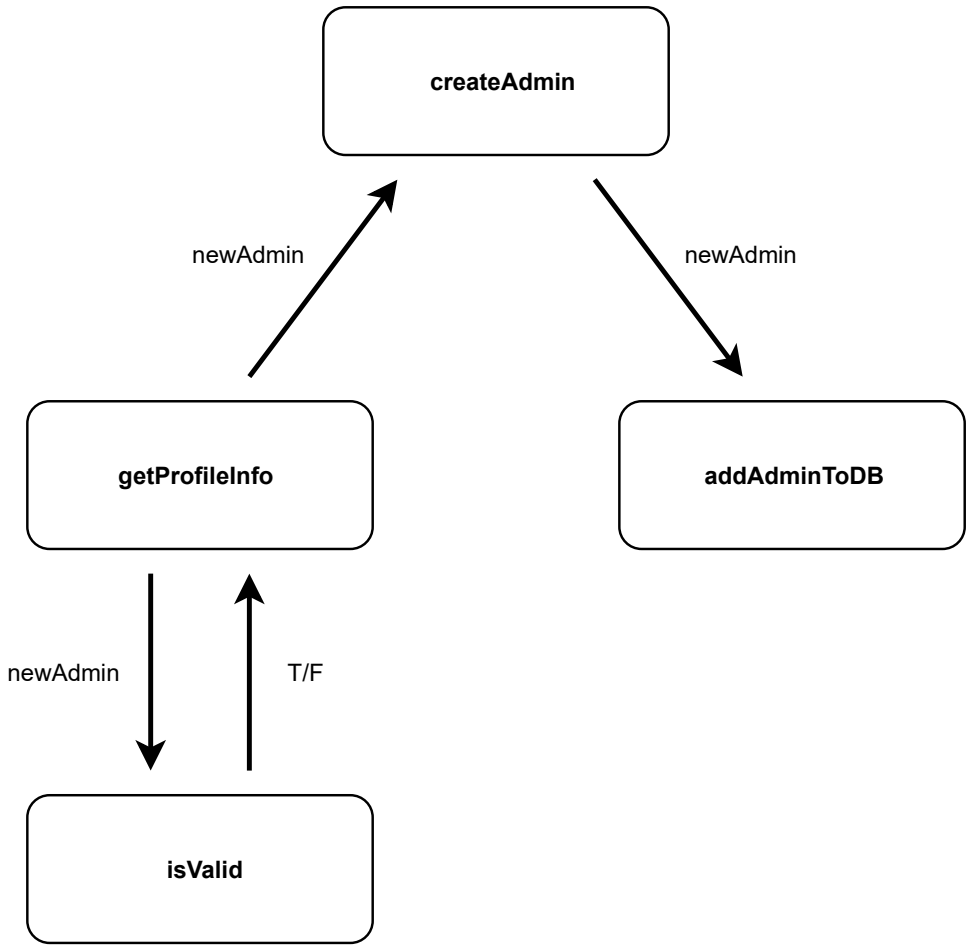
Name	1.2.2.1.15.1 Auction Class Diagram
Purpose:	To diagram the attributes and methods of the auction class and how it relates to other classes.
Description:	The auction class diagram will show attributes, methods and relationships with other classes through dependency.
Requirements:	PO.2.2.17, FS.23, FS.23.4, FV.23, FV.23.4
Elements:	1.2.2.1.15.2 Bid Class Diagram
Referenced By:	1.2.2.1 Web Client UI
Viewpoint:	Class Diagram

View 1.2.2.1.15.2 Bid Class



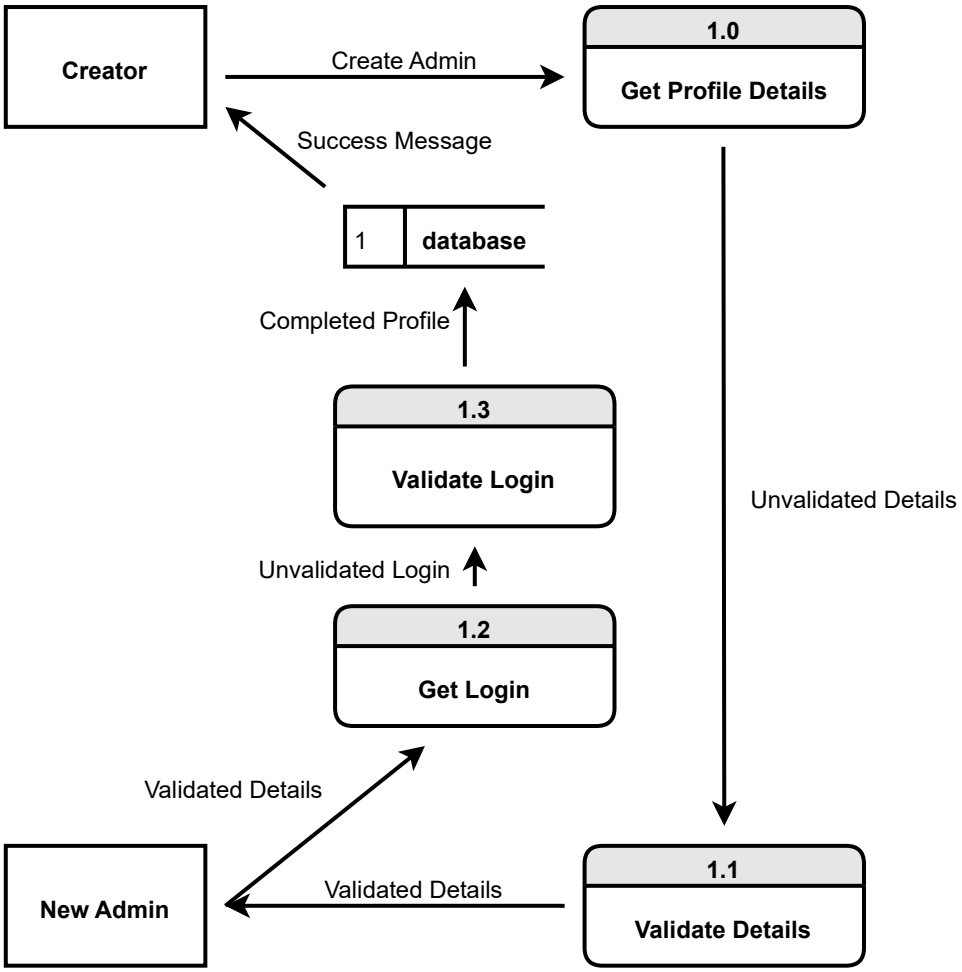
Name:	**1.2.2.1.15.2 Bid Class**
Purpose:	Describe the details of the Bid class.
Description:	This diagram shows the attributes and methods that belong to the Bid class.
Requirements:	FS.23.4
Elements:	1.2.2.1.15.1 Auction Class
Referenced By:	1.2.2.1.15 Web Client Auction Page
Viewpoint	Class Diagram

View 1.2.4.1: Create Admin



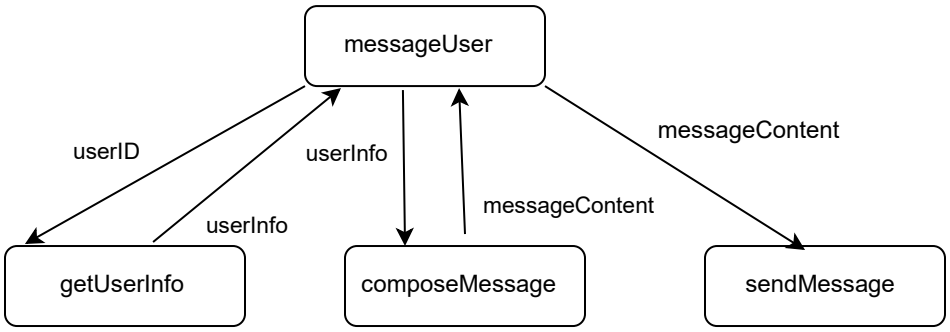
Name	1.2.4.1: Create Admin
Purpose	This will describe the components pertaining to the admin's ability to create other admins.
Description	This will show the connection between the admin page and how admins are created.
Requirements	[PO 1.7.32], [PO 2.3.6]
Elements	**1.2.4.1.1 setProfileInfo**
	1.2.4.1.2 isValid
	1.2.4.1.3 addAdminToDB
Referenced By	None yet
Viewpoint	Structure Chart

View 1.2.4.1.1: Set Profile Info



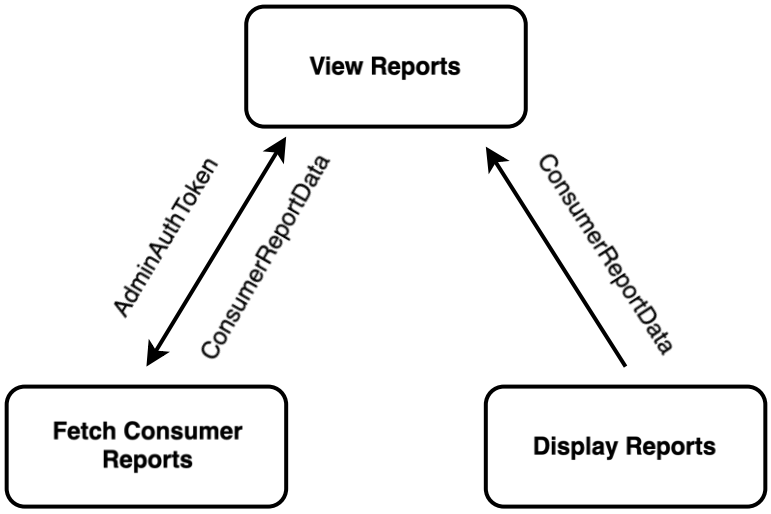
Name	1.2.4.1.1: Set Profile Info
Purpose	This describes the component related to setting information in a new admin's profile.
Description	This shows how the data related to creating an admin profile is created and validated throughout the setProfileInfo process.
Requirements	[PO.1.7.32], [PO.2.3.6]
Elements	**1.2.4.1.2 Get Profile Details**
	1.2.4.1.3 Validate Details
	1.2.4.1.4 Get Login
	1.2.4.1.5 Validate Login
Referenced By	#1.2.4.1
Viewpoint	Data Flow Diagram

View 1.2.4.2: Message



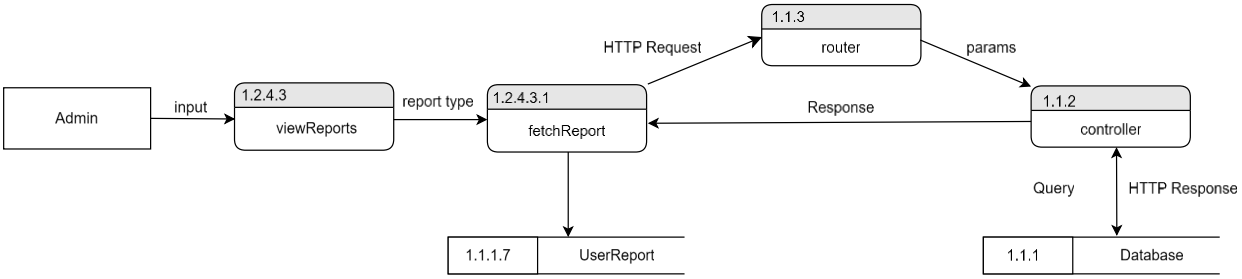
Name	**1.2.4.2: Message**
Purpose:	This view describes the components pertaining to the Admin's ability to message any type of user (commercial or consumer) from the admin page.
Description:	This view will show the connection between the admin page and the messaging functionality, illustrating how an admin can send messages to commercial and consumer users.
Requirements:	[P.O. 1.7.36], [P.O. 2.3.9]
Elements:	**1.2.4.2.1 messageUser:** This function will allow the admin to initiate the messaging process.
	1.2.4.2.2 getUserInfo: This function will retrieve user details needed to send a message.
	1.2.4.2.3 composeMessage: This function will enable the admin to compose the message
	1.2.4.2.4 sendMessage: This function will handle sending the composed message to the selected user.
Referenced By:	None yet
Viewpoint:	Structure Chart

View 1.2.4.3: View Reports



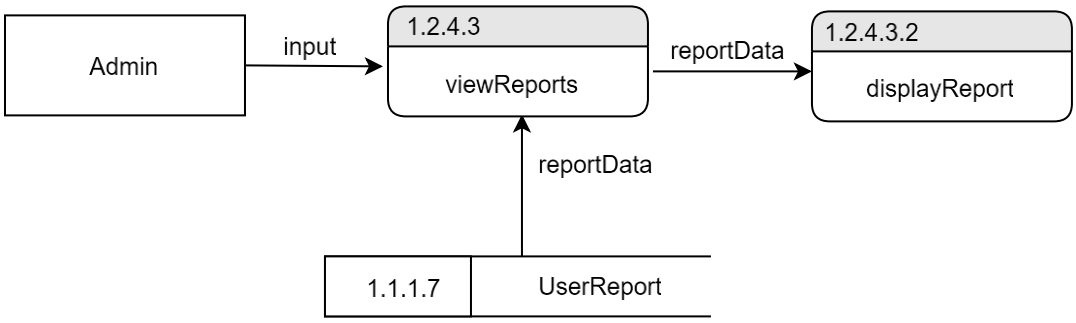
Name	1.2.4.3 View Reports
Purpose:	Describes the functions used to view consumer or commercial reports from the admin page.
Description:	Describes the individual functions required for the admin to be able to view consumer or commercial reports.
Requirements:	[P.O 2.3.1], [P.O. 2.3.2]
Elements:	1.2.4.3.1 **fetchReports –** A function that fetches all consumer or commercial reports for the user
	1.2.4.3.2 **displayReport –** A function that displays the report data for the user
Referenced By:	1.2.4 Admin Console
Viewpoint:	Structure Chart

View 1.2.4.3.1 Fetch Reports



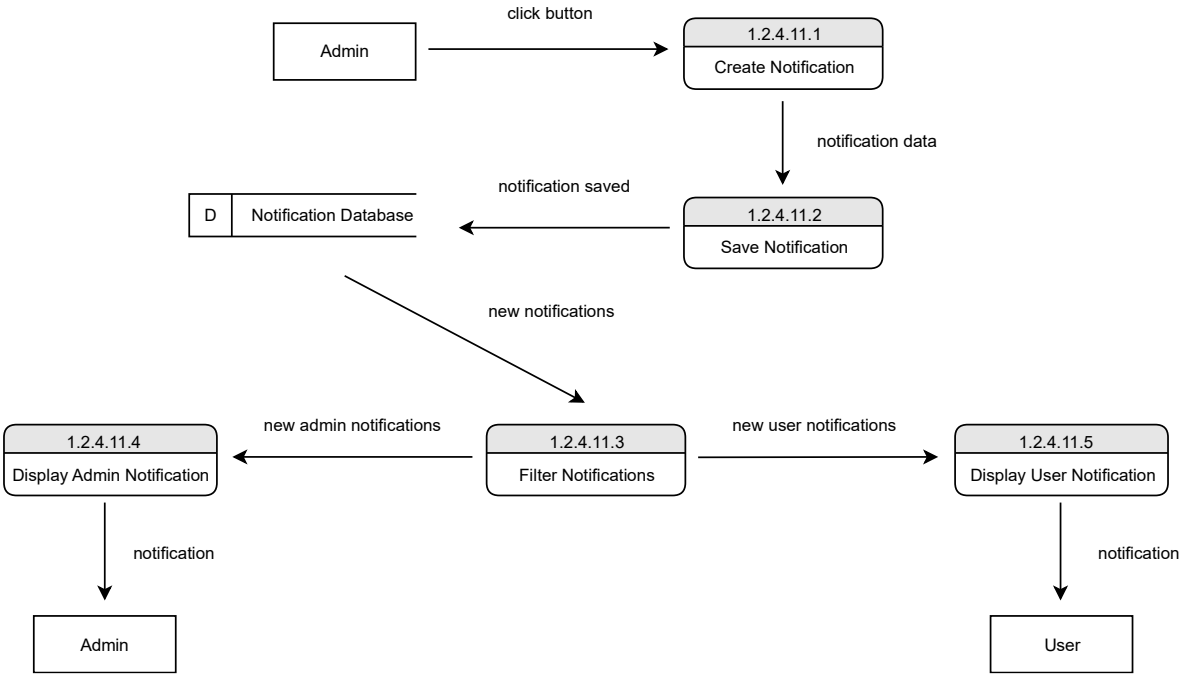
Name	1.2.4.3.1 Fetch Reports
Purpose:	Describes the components related to fetching consumer reports from the database.
Description:	Shows the data flow between when the admin user presses a 'view reports' button and when it fetches the report using the viewReports function.
Requirements:	[P.O 2.3.1], [P.O. 1.2.4.1]
Elements:	**1.2.4.3 viewReports –** The initial function that is called when the user presses the 'view reports' button on the admin event page.
	1.2.4.3.3 reportData – A variable inside of the viewReports function that stores the data from the fetchReport call.
	1.1.1 Database – Stores the data
	1.1.2 Controller – Handles the request interaction with the server
	1.1.3 Router – Routes server requests for the API.
Referenced By:	1.2.4.3 viewReports
Viewpoint:	Data Flow Diagram

View 1.2.4.3.2 Display Report



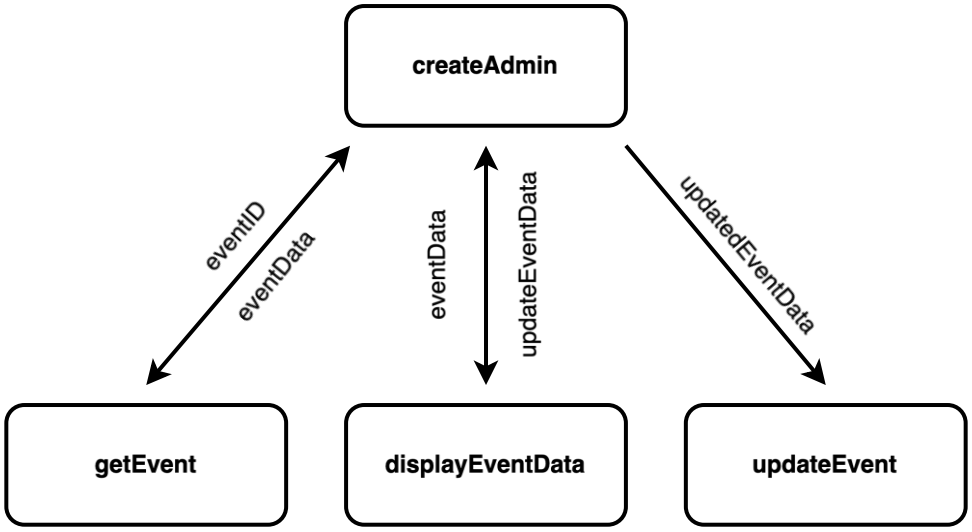
Name	**1.2.4.3.2 Display Report**
Purpose:	Describe the components related to displaying consumer report data for the admin user.
Description:	Describes the connection between the admin interactions and the functions to display the fetched consumer report data.
Requirements:	[P.O 2.3.1], [P.O. 2.3.2]
Elements:	**1.2.4.3 Fetch Report –** Describes the initial function that is called when the admin interacts with the 'view report' button on the admin screen.
	1.2.4.3.3 Report Data – A variable inside of the viewReports function that stores the data from the fetchReport call.
Referenced By:	1.2.4.3 View Reports
Viewpoint:	Data Flow Diagram

View 1.2.4.4: Notifications



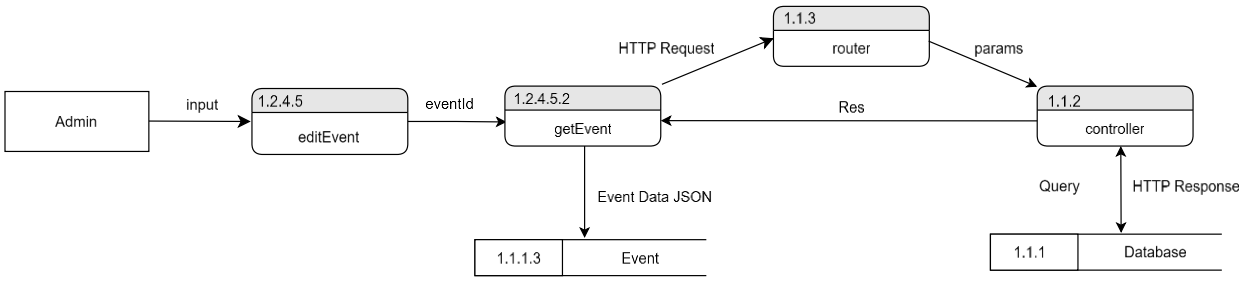
Name	1.2.4.4 Notifications
Purpose:	Describe the admin's ability to notify users or other admins of issues that need to be addressed.
Description:	Show the connection between the admin page and how admins will send/receive notifications.
Requirements:	[PO.1.7.41], [PO.2.3.11]
Elements:	**1.2.4.4.1 Create Notification:** Process by which the administrator creates a new notification.
	1.2.4.4.2 Save Notification: Process of saving the created notification in the database.
	1.2.4.4.3 Filter Notifications: Process that filters new notifications to determine whether they are for users or administrators.
	1.2.4.4.4 Display Admin Notification: Process that displays new notifications to administrators.
	1.2.4.4.5 Display User Notification: Process that displays new notifications to users.
Referenced By:	1.2.4 Admin
Viewpoint:	Data Flow Diagram

View 1.2.4.5: Edit Events



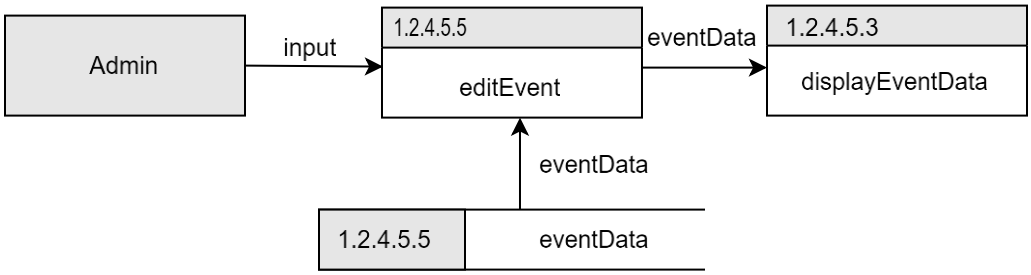
Name	1.2.4.5: Edit Events
Purpose	This view describes the functions used to edit any event from the admin page.
Description	This view describes the individual components and functions relating to the admins ability to edit their profile.
Requirements	[PO 1.7.38], [PO 2.3.9]
Elements	**1.2.4.5.1 EditEvent:** This function will allow the admin to prompt the start of the editEvent process. This will be used in the form of a button inside of the events page, which will only be rendered if the user viewing it is an admin.
	1.2.4.5.2 **getEvent:** This function will use the eventId to query the database for the relevant information.
	1.2.4.5.3. displayEventData: This function will take the eventData received from getEvent to render the eventData in a way that will let the user edit it.
	1.2.4.5.4 updateEvent: After the data is edited, the updateEvent function will be called create an update request to the database with the new event data.
Referenced By	1.2.4
Viewpoint	Structure Chart

View 1.2.4.5.2: Get Event view



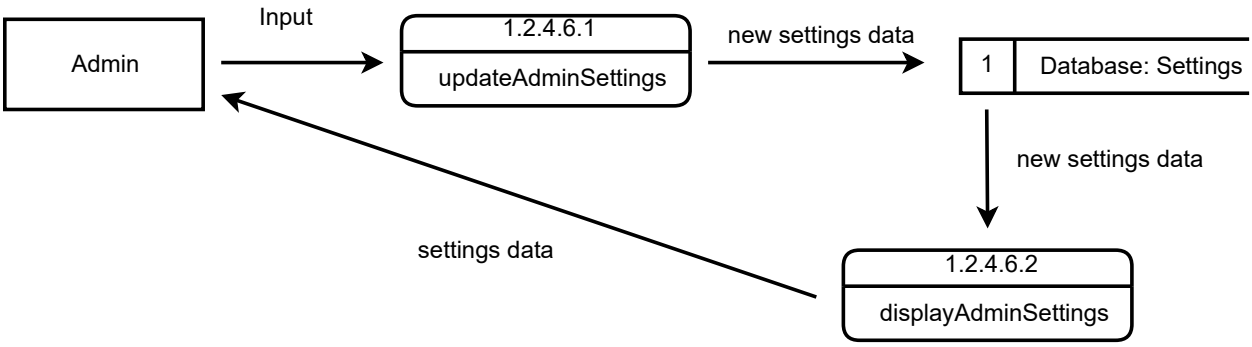
Name	1.2.4.5.2 Get Event View
Purpose:	Describes the components related to fetching specific event data from the database using a specific event ID.
Description:	Shows the connection between the admin page and the function to query the database for the event information using the eventide.
Requirements:	[P.O. 1.7.38], [P.O. 2.3.9]
Elements:	**1.1.1 Database** – Holds the data
	1.1.2 Controller – Handles request interaction with the server
	1.1.3 Router – Routes server requests for the API.
	1.2.4.5.5 eventData – A variable defined inside of the editEvent function to hold the response from the getEvent function.
Referenced By:	1.2.4.5 Edit Events
Viewpoint:	Data Flow Diagram

View 1.2.4.5.3: Display Event Data



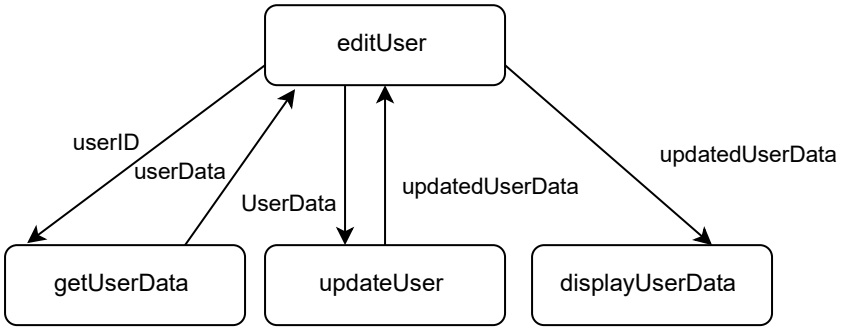
Name	**1.2.4.5.3 Display Event Data**
Purpose:	Describes the components related to displaying event data for the user.
Description:	Shows the data flow between the admin page and how editEvent renders data for the user to edit.
Requirements:	[P.O 1.7.38], [P.O. 2.3.9]
Elements:	**1.2.4.5 editEvent –** The initial function that is called when the user presses the 'edit event' button on the admin event page.
	1.2.4.5.5 eventData – A variable inside of the editEvent function that stores the 1.2.4.5.2 - getEvent response.
Referenced By:	1.2.4.6 editEvent
Viewpoint:	Data Flow Diagram

View 1.2.4.6: Settings



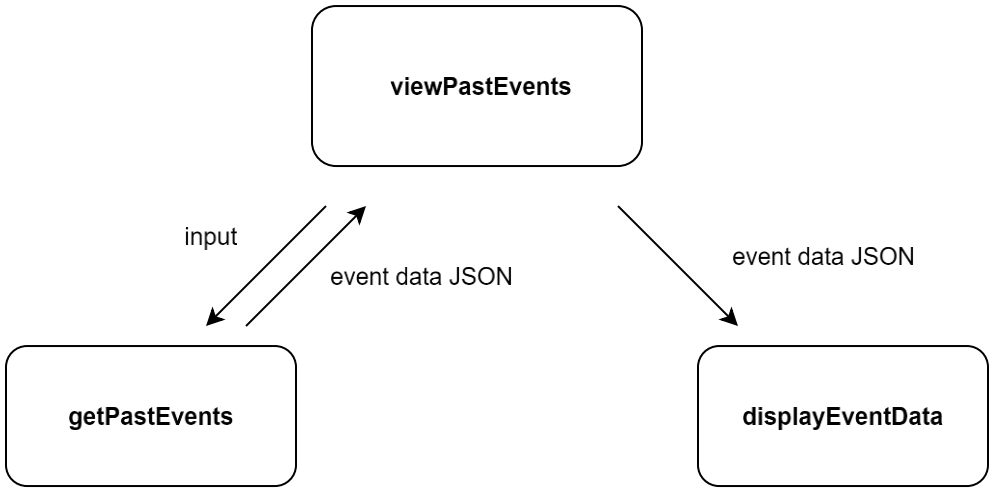
Name	1.2.4.6 Settings
Purpose:	Describe the components pertaining to the admin's ability to customize their own settings.
Description:	Show the connection between the admin and how they can customize their own settings.
Requirements:	[P.O.1.7.37], [P.O. 1.7.42], [P.O.2.3.12]
Elements:	**1.2.4.6.1 updateAdminSettings:** Processes and saves changes made by the administrator to your personal settings.
	1.2.4.6.2 displayAdminSettings: Retrieves and displays the current administrator settings from the database.
Referenced By:	1.2.4 Admin
Viewpoint:	Data Flow Diagram

View 1.2.4.7: Edit Users



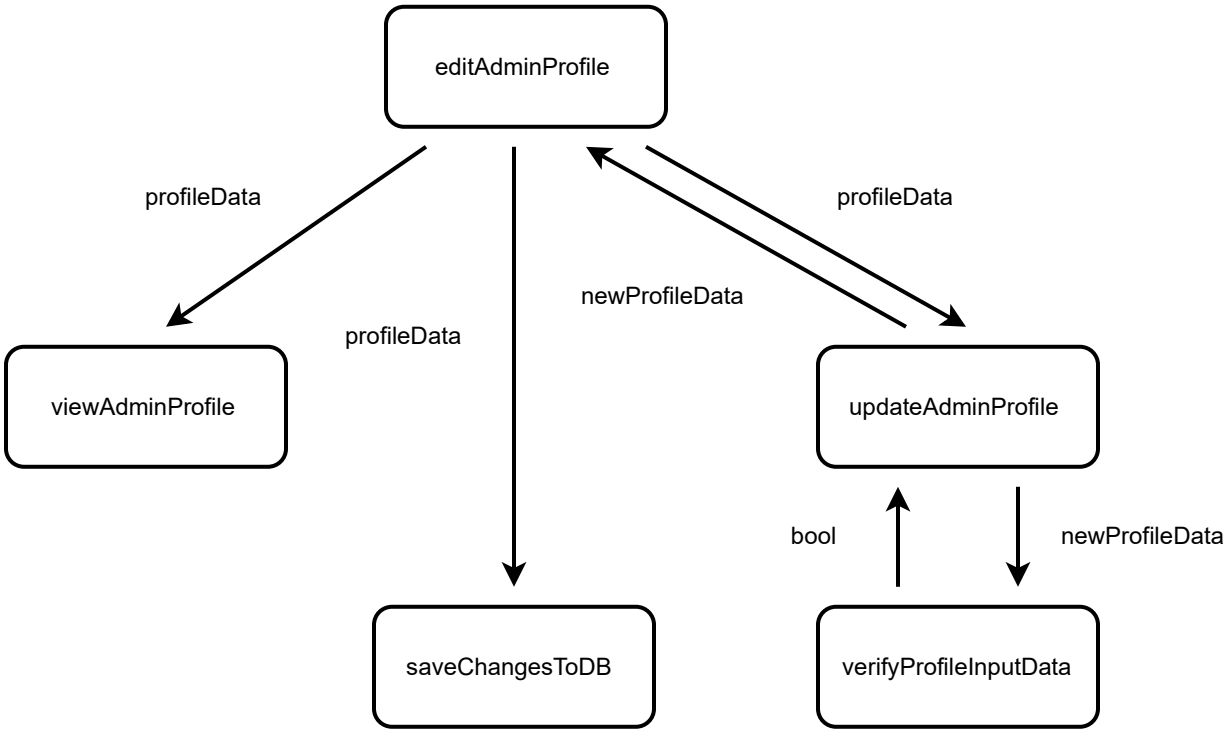
Name	**1.2.4.7: Edit Users**
Purpose:	This view describes the functions used to edit any user (commercial or consumer) from the admin page.
Description:	This view describes the individual components and functions relating to the admin's ability to edit user profiles.
Requirements:	[PO 1.2.18], [PO 1.7.38]
Elements:	**1.2.4.7.1 editUser:** This function will allow the admin to prompt the start of the editUser process. This will be used in the form of a button inside of the user management page, which will only be rendered if the user viewing it is an admin.
	1.2.4.7.2 getUserData: This function will use the userId to query the database for the relevant information
	1.2.4.7.3 displayUserData: This function will take the userData received from getUserData to render the userData in a way that will let the user edit it.
	1.2.4.7.4 updateUser: After the data is edited, the updateUser function will be called to create an update request to the database with the new user data.
Referenced By:	None yet
Viewpoint:	Structure Chart

View 1.2.4.8 : Past Events



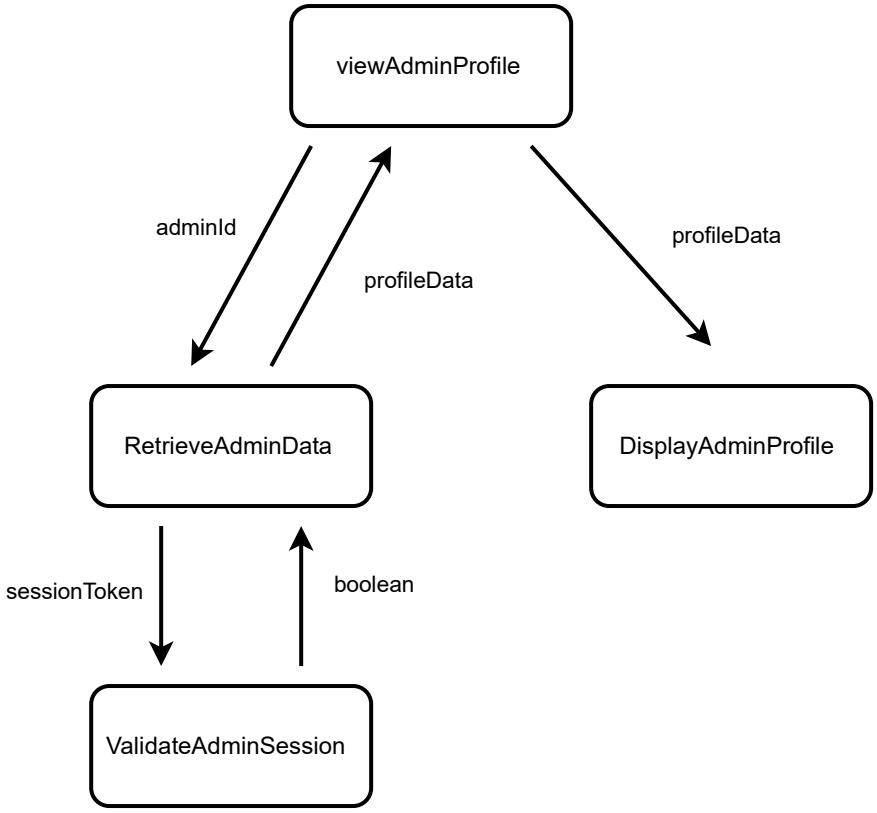
Name	1.2.4.8 Past Events
Purpose:	Illustrates the admin's ability to see past events.
Description:	Illustrates the functionality that lets the admin see past events using the viewPastEvents function.
Requirements:	PO 1.7.35 The system shall allow administrators to read, archive, copy and edit past events.
Elements:	**1.2.4.8.1 getPastEvents –** Shows the how the Admin would get past events.
	1.2.4.5.3 displayEventData – Shows how the data would be displayed to the admin after it is received.
Referenced By:	No references.
Viewpoint:	Component Diagram

View 1.2.4.9: Edit Profile



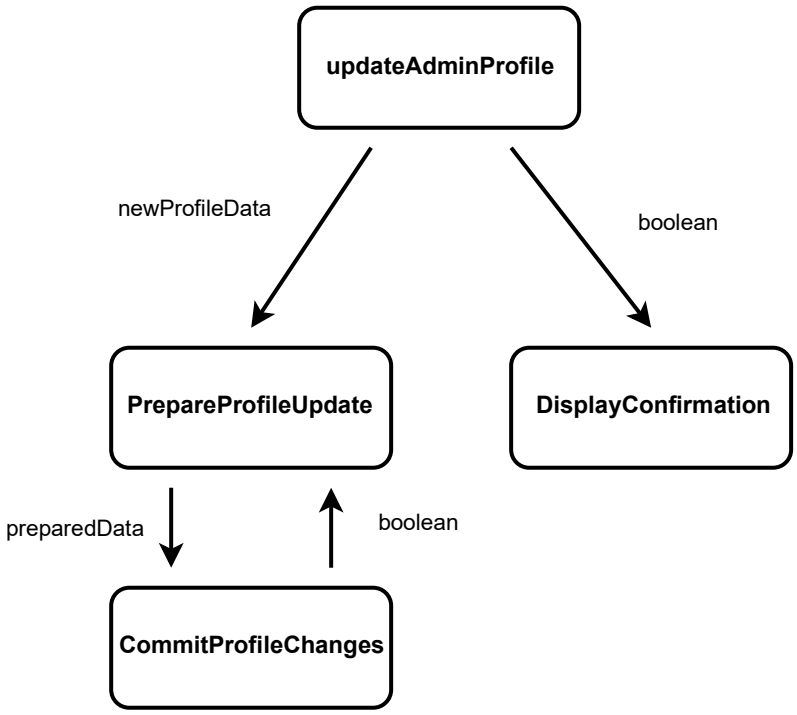
Name	1.2.4.9: Edit Profile
Purpose	The admin will have the option to customize their profile in a variety of ways, just like a normal user. This will describe their ability to do that.
Description	This will describe the components pertaining to the admins ability to customize and edit their profile.
Requirements	[PO 1.7.37], [PO 2.3.3], [PO 2.3.4]
Elements	1.2.4.9.1 **viewAdminProfile**: This function allows the admin to view their current profile.
	1.2.4.9.2**: updateAdminProfile:** This function enables the admin to update their profile information.
	1.2.4.9.3**: verifyProfileInputData**: This function verifies the input data provided by the admin before updating the profile.
	1.2.4.9.4**: saveChangesToDB**: This function saves the changes made to the admin's profile in the database.
Referenced By	1.2.4
Viewpoint	Structure Chart

View 1.2.4.9.1: viewAdminProfile



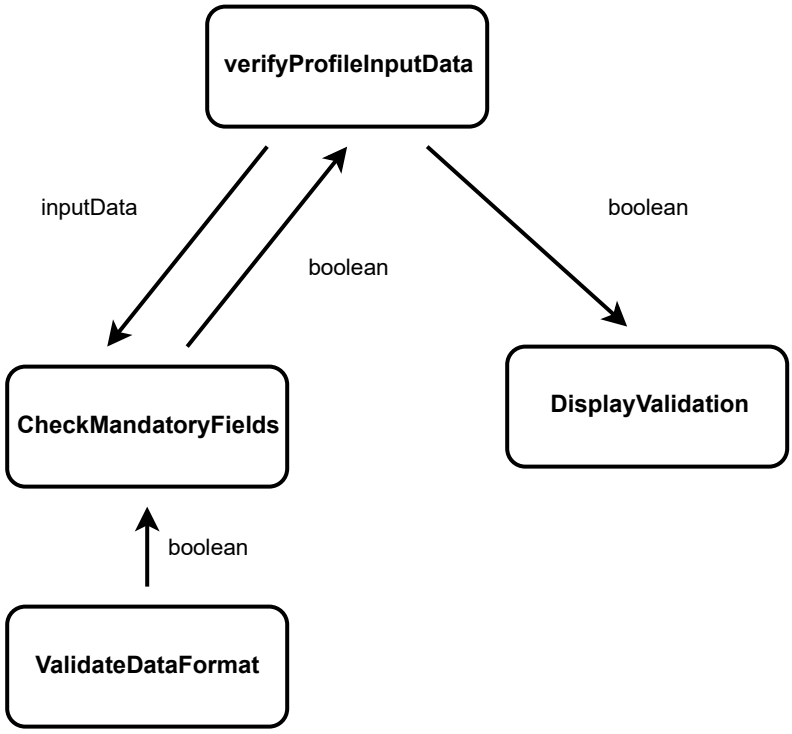
Name	1.2.4.9.1 viewAdminProfile
Purpose:	Describe the components related to the admin's ability to view their current profile.
Description:	Show the connection between the admin page and the function to display the admin's current profile information.
Requirements:	[PO 1.7.37], [PO 2.3.3], [PO 2.3.4]
Elements:	**1.2.4.9.1.1 RetrieveAdminData:** Retrieves administrator profile data from the database. **1.2.4.9.1.2 ValidateAdminData:** Validates that the administrator session is active and valid. **1.2.4.9.1.3 DisplayAdminProfile:** Displays the administrator profile information on the user interface.
Referenced By:	1.2.4.9 Edit Admin Profile
Viewpoint:	Structure Chart

View 1.2.4.9.2: updateAdminProfile



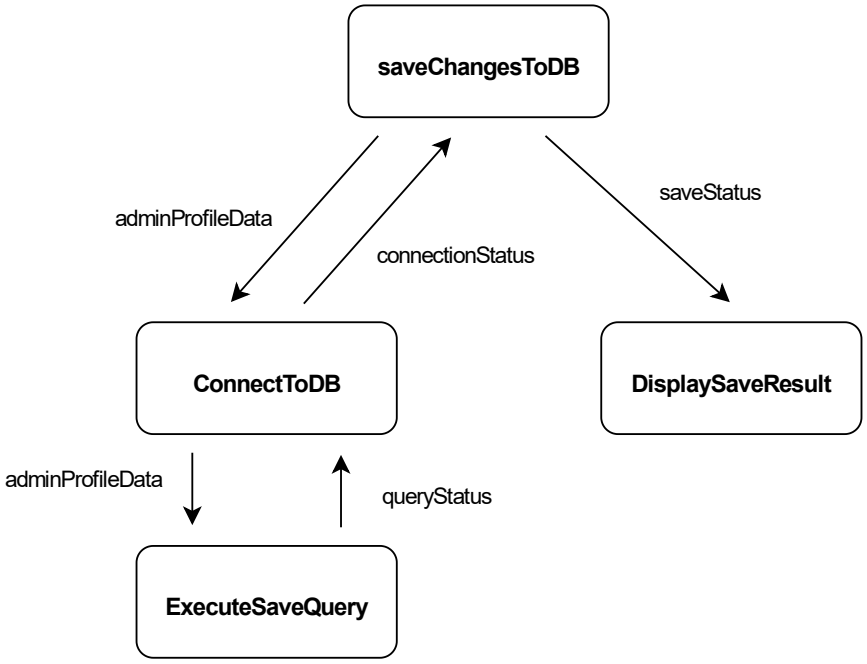
Name	1.2.4.9.2 updateAdminProfile
Purpose:	Describe the components related to the admin's ability to update their profile information.
Description:	Show the connection between the admin page and the function to update the admin's profile information.
Requirements:	[PO 1.7.37], [PO 2.3.3], [PO 2.3.4]
Elements:	**1.2.4.9.2.1 PrepareProfileUpdate:** Prepare the profile data to be updated in the database. **1.2.4.9.2.2 CommitProfileChanges:** Makes the profile changes to the database. **1.2.4.9.2.3 DisplayConfirmation:** Displays the update confirmation in the UI.
Referenced By:	1.2.4.9 Edit Admin Profile
Viewpoint:	Structure Chart

View 1.2.4.9.3: verifyProfileInputData



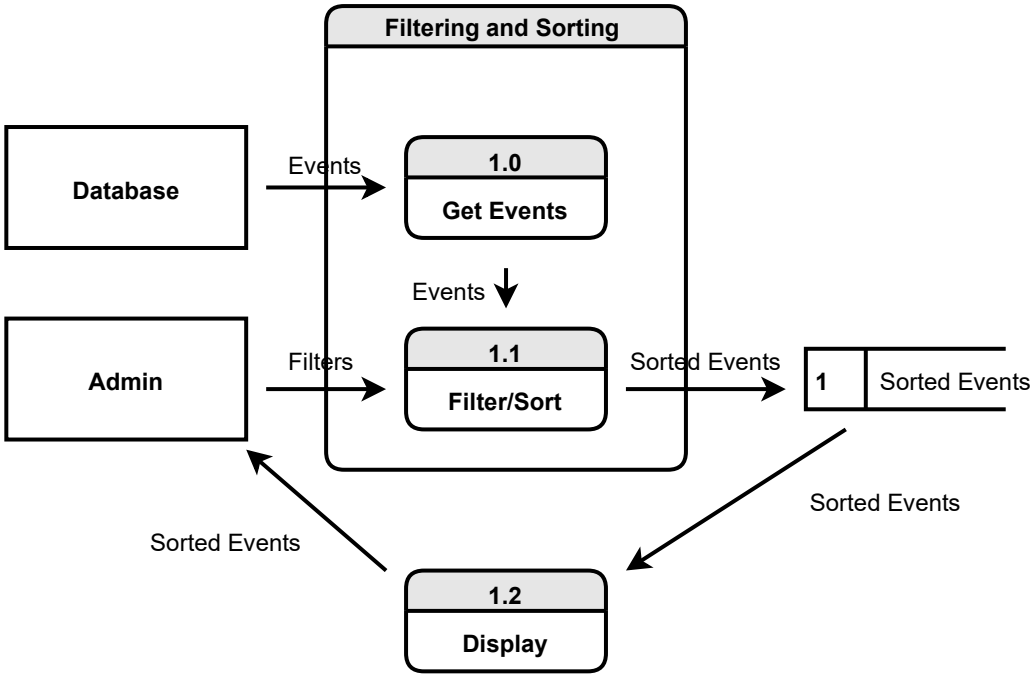
Name	1.2.4.9.3 verifyProfileInputData
Purpose:	Describe the components related to verify the input data provided by the admin before updating the profile.
Description:	Show the connection between the admin page and the function to validate the profile input data.
Requirements:	[PO 1.7.37], [PO 2.3.3], [PO 2.3.4]
Elements:	**1.2.4.9.3.1 CheckMandatoryFields:** Checks that all required fields are present. **1.2.4.9.3.2 ValidateDataFormat:** Validates the format of the entered data. **1.2.4.9.3.3 DisplayValidation:** Displays validation errors in the user interface.
Referenced By:	1.2.4.9 Edit Admin Profile
Viewpoint:	Structure Chart

View 1.2.4.9.4: saveChangesToDB



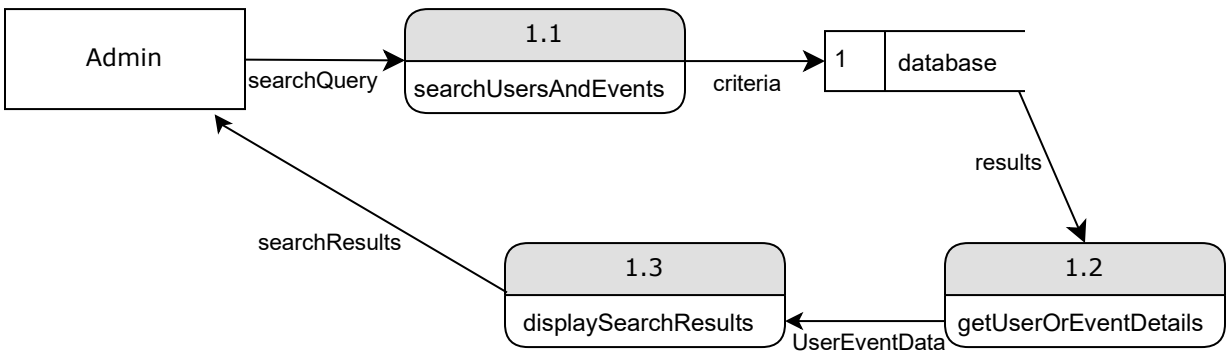
Name	1.2.4.9.4 saveChangesToDB
Purpose:	Describe the components related to saving the changes made to the admin's profile in the database.
Description:	Show the connection between the admin page and the function to save the updated profile information in the database.
Requirements:	[PO 1.7.37], [PO 2.3.3], [PO 2.3.4]
Elements:	**1.2.4.9.4.1 ConnectToDB:** Establishes a connection to the database. **1.2.4.9.4.2 ExecuteSaveQuery:** Execute the query to save the profile changes. **1.2.4.9.4.3 DisplaySaveResult:** Displays the result of saving changes to the user interface.
Referenced By:	1.2.4.9 Edit Admin Profile
Viewpoint:	Structure Chart

View 1.2.4.10: Filter / Sorting



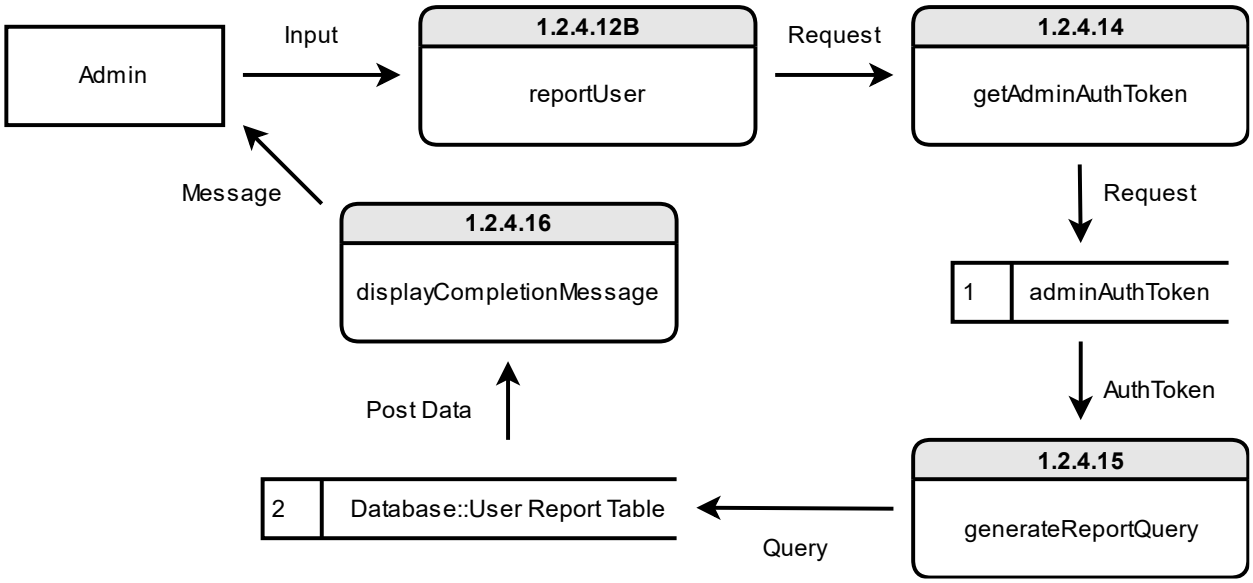
Name	**1.2.4.10: Filter / Sorting**
Purpose:	This view describes the components and interactions related to the admin's ability to filter and sort users and events within the UEvents system for user management purposes. The main goal is to facilitate the management and organization of user and event data by an administrator.
Description:	The Filtering and Sorting view illustrates the connection between the admin page (Admin Console) and the filtering and sorting functionality. This subsystem allows administrators to filter and sort users and events and also communicates with the database to retrieve and update the filtered and sorted data.
Requirements:	[P.O. 1.7.38], [P.O. 1.7.39]
Elements:	**1.2.2.1.1.1 Admin Console:** Interface through which administrators interact with the system.
	1.2.2.1.1. 2 Filtering and Sorting Subsystem: Handles the logic for filtering and sorting functionality.
	1.2.2.1.1. 3 Database: Stores user and event data and records updates.
Referenced By:	None yet
Viewpoint:	Data Flow Diagram

View 1.2.4.11: Search Bar



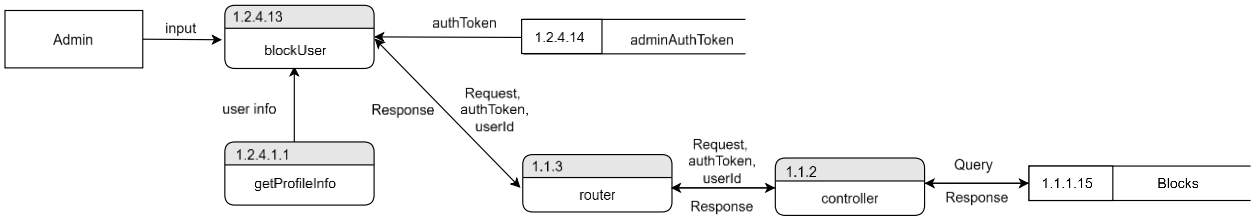
Name	1.2.4.11: Search Bar
Purpose:	Provides the admin with the ability to search for specific users or events.
Description:	Shows the integration and functionality of the search bar in the admin interface, allowing the admin to efficiently locate specific users or events by entering relevant keywords or identifiers.
Requirements:	[PO.1.7.40]
Elements:	**1.2.4.11.1 searchUsersAndEvents:** This function will allow the admin to initiate a search for users or events.
	1.2.4.11.2 getUserOrEventDetails: This function will retrieve the details of users or events based on the search query.
	1.2.4.11.3 displaySearchResults: This function will display the search results to the admin.
Referenced By:	1.2.4
Viewpoint:	Data Flow Diagram (DFD)

View 1.2.4.12: Report User



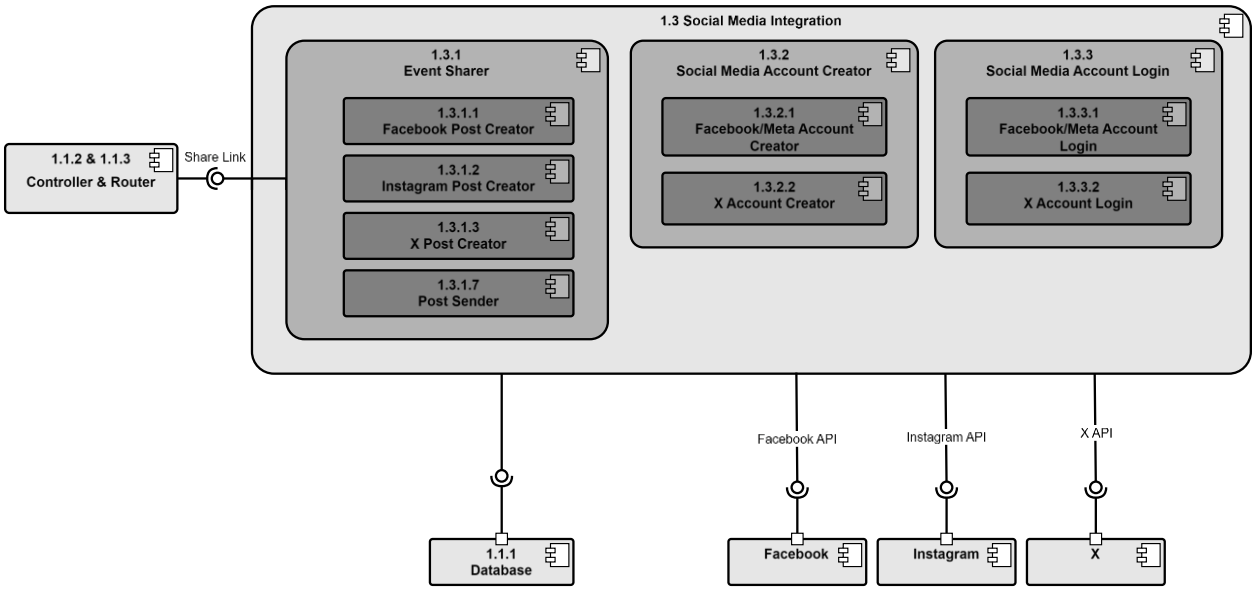
Name	**1.2.4.12 Report User**
Purpose	Allows admins to report users when viewing their profiles.
Description	Shows the functionality and integration of the "Report User" option in consumer and commercial profiles.
Requirements	[FS.40.1], [FS.39.1]
Elements	**1.2.4.12B reportUser**
	1.2.4.14 getAdminAuthToken
	1 adminAuthToken
	1.2.4.15 generateReportQuery
	2 User Report Table
	1.2.4.16 displayCompletionMessage
Referenced by	1.2.4
Viewpoint	Data Flow Diagram

View 1.2.4.13 Block User



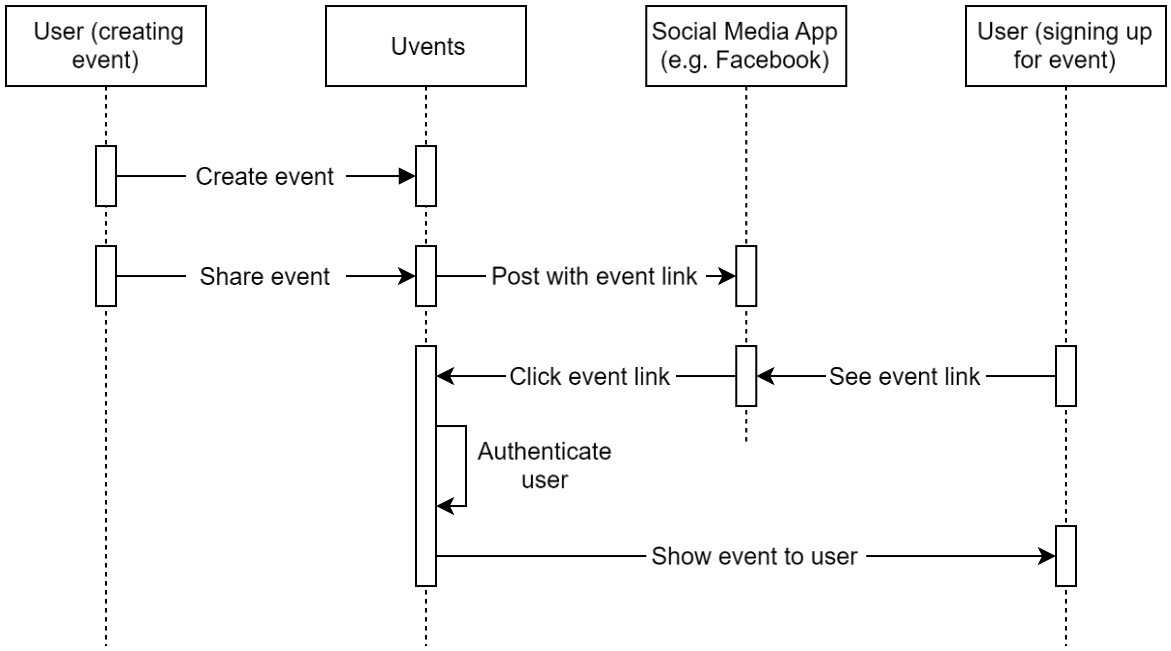
Name	1.2.4.13 Block User
Purpose:	Allows Admins to block users when viewing their profiles.
Description:	Shows the functionality and integration of the "Block User" option that appears if the user is an Admin while viewing consumer and commercial profiles.
Requirements:	[FS.39.2], [FS.40.2]
Elements:	**1.2.4.1.1 getProfileInfo –** Gets the information of a specific user
	1.2.4.3.3 adminAuthToken – The adminAuthToken is the token that verifies the user performing specific functions is an Admin. This token is stored in the localstorage
	1.1.3 Router – Routes server requests for the API.
	1.1.2 Controller – Handles the request interaction with the server.
	1.1.1.15 Blocks – Stores the information of blocked users
Referenced By:	1.2.4.6 editEvent
Viewpoint:	Data Flow Diagram

View 1.3: Social Media Integration



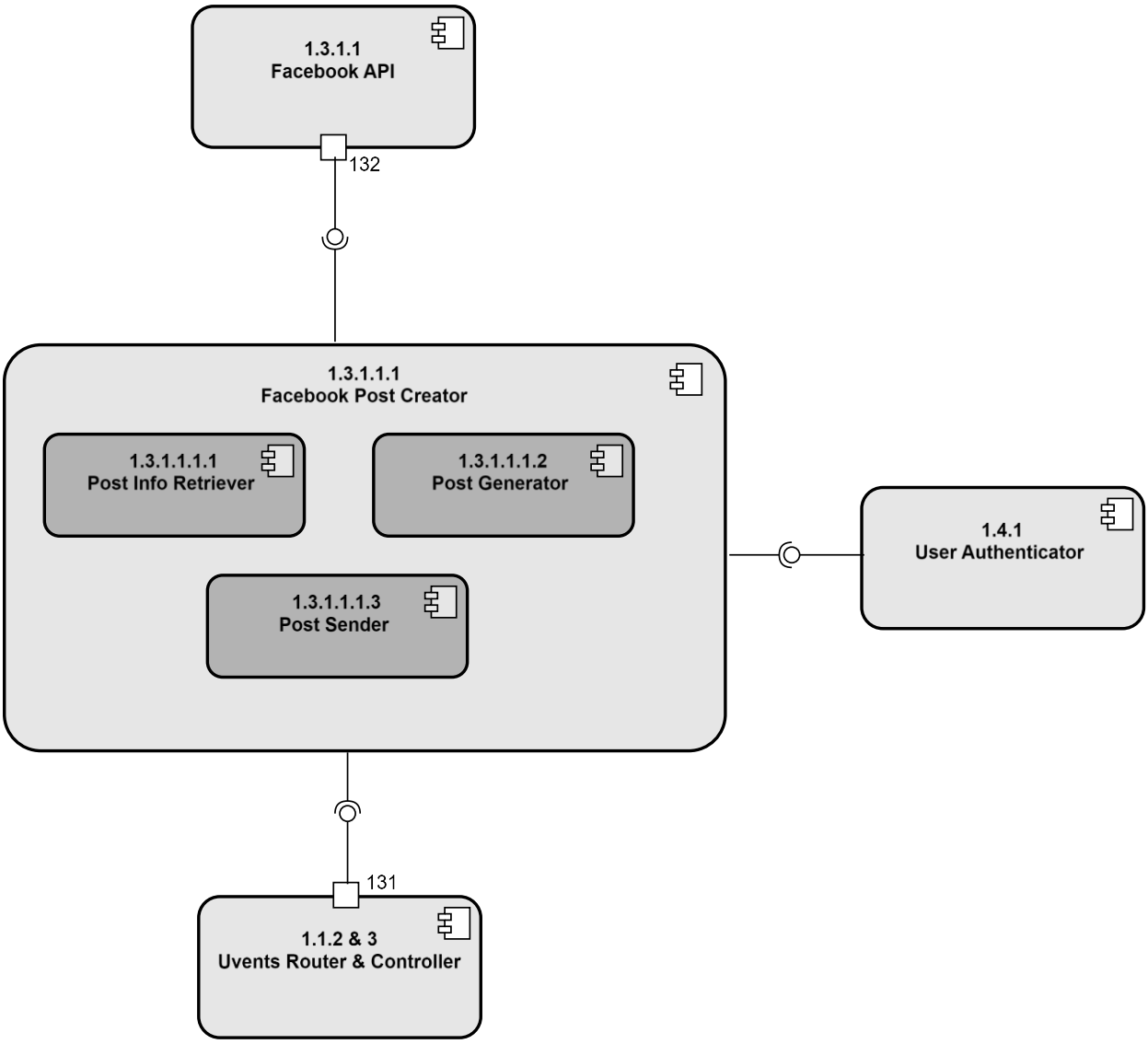
Name	1.3 Social Media Integration
Purpose:	Illustrates the component for integrating with social media.
Description:	Illustrates all the components used to interact with third party social media platforms.
Requirements:	EIS.2 Facebook [PO.1.4.1], EIS.3 Instagram [PO.1.4.1], EIS.4 Twitter [PO.1.4.1], FS.1.4 [LDR.1.6], FS.23.5.3 [PO.2.2.18]
Elements:	**1.3.1.1 Event Sharer**
	1.3.1.2 Social Media Account Creator
	1.3.1.3 Social Media Account Login
Referenced By:	1
Viewpoint:	Component Diagram

View 1.3.1: Sharing Events Sequence Diagram (User)



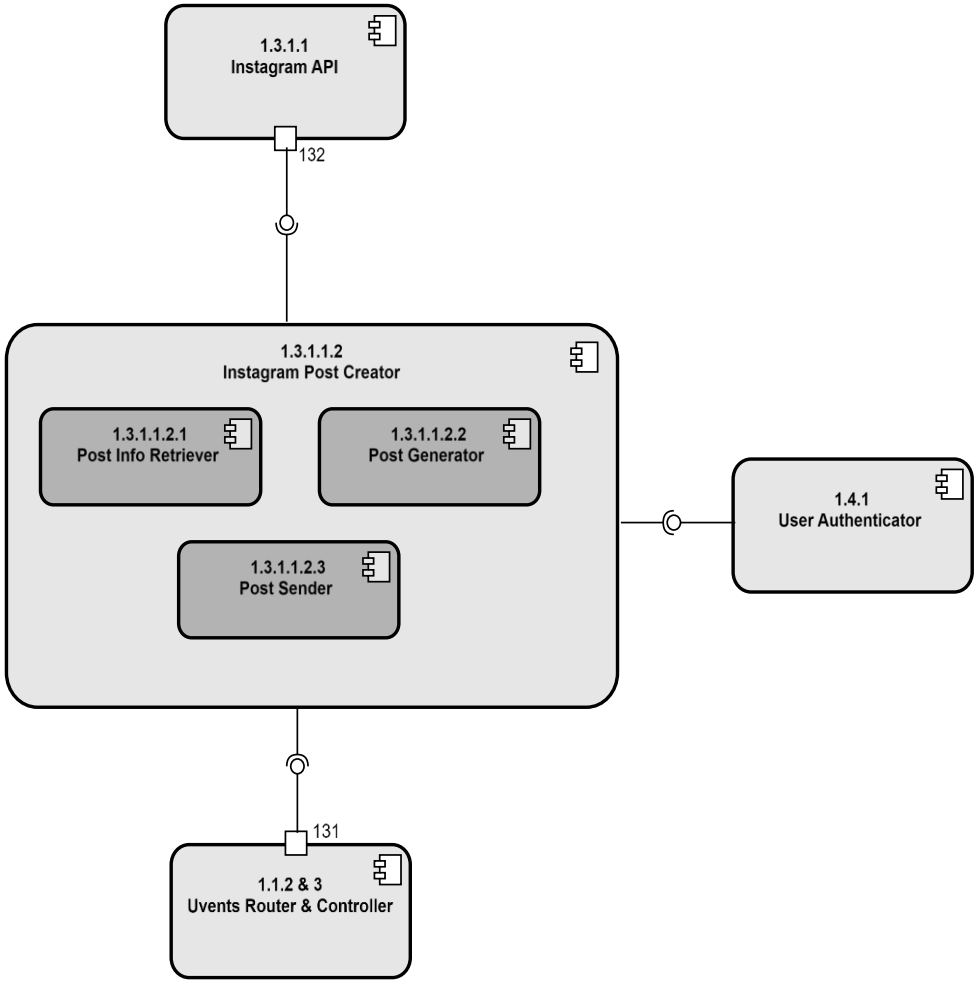
Name	**1.3.1: Sharing Events Sequence Diagram (User)**
Purpose	A view of the sharing workflow.
Description	This diagram shows how users will share events and respond chronologically.
Requirements	N/A
Elements	**1.3.1.4 Share Link Generator**
	1.3.1.5 Share Link Navigation
Referenced by	1.3.1
Viewpoint	Sequence Diagram

View 1.3.1.1: Facebook Post Creator



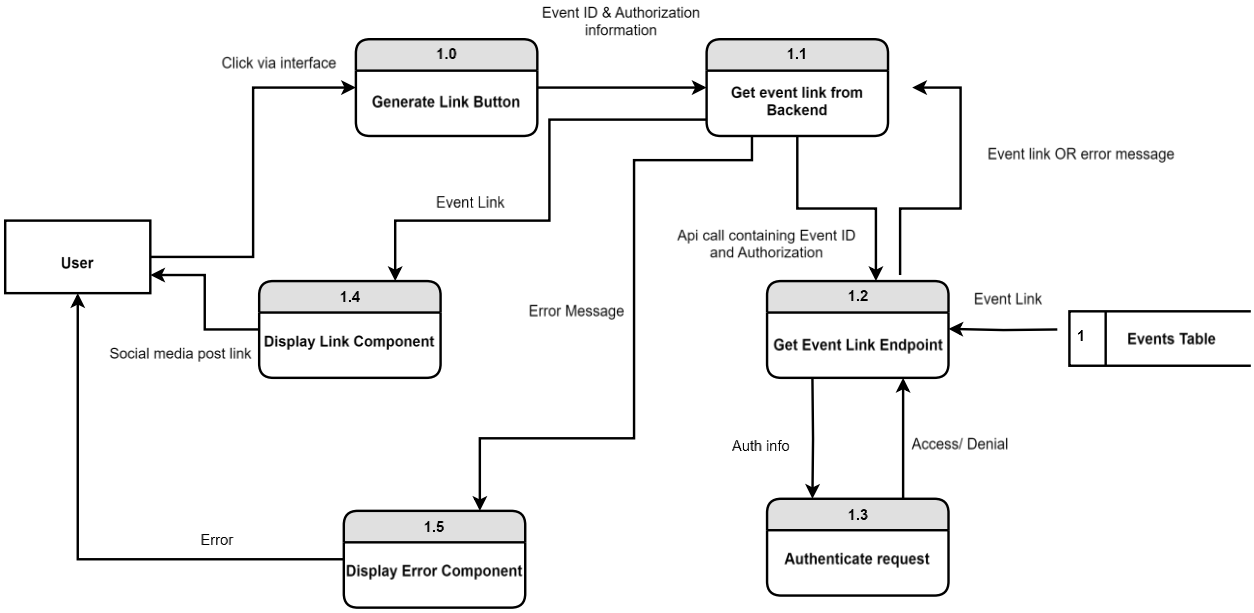
Name	**1.3.1.1 Facebook Post Creator**
Purpose:	Illustrates the component for Facebook post creation
Description:	Illustrates how/where Facebook posts are created in the system and how information is obtained and formatted and how a post is created
Requirements:	EIS.2, EIS.4.1 [PO.1.4.1]
Elements	**1.3.1.1 Post Info Retriever:** fetches the post and user info
	1.3.1.2 Post Generator: formats/generates the post
	1.3.1.3 Post Sender: sends the generated post to be displayed
	1.1.2 & 3 Uvents Router & Controller
Referenced By:	1.3 Social Media Integration
Viewpoint:	Component Diagram

View 1.3.1.2: Instagram Post Creator



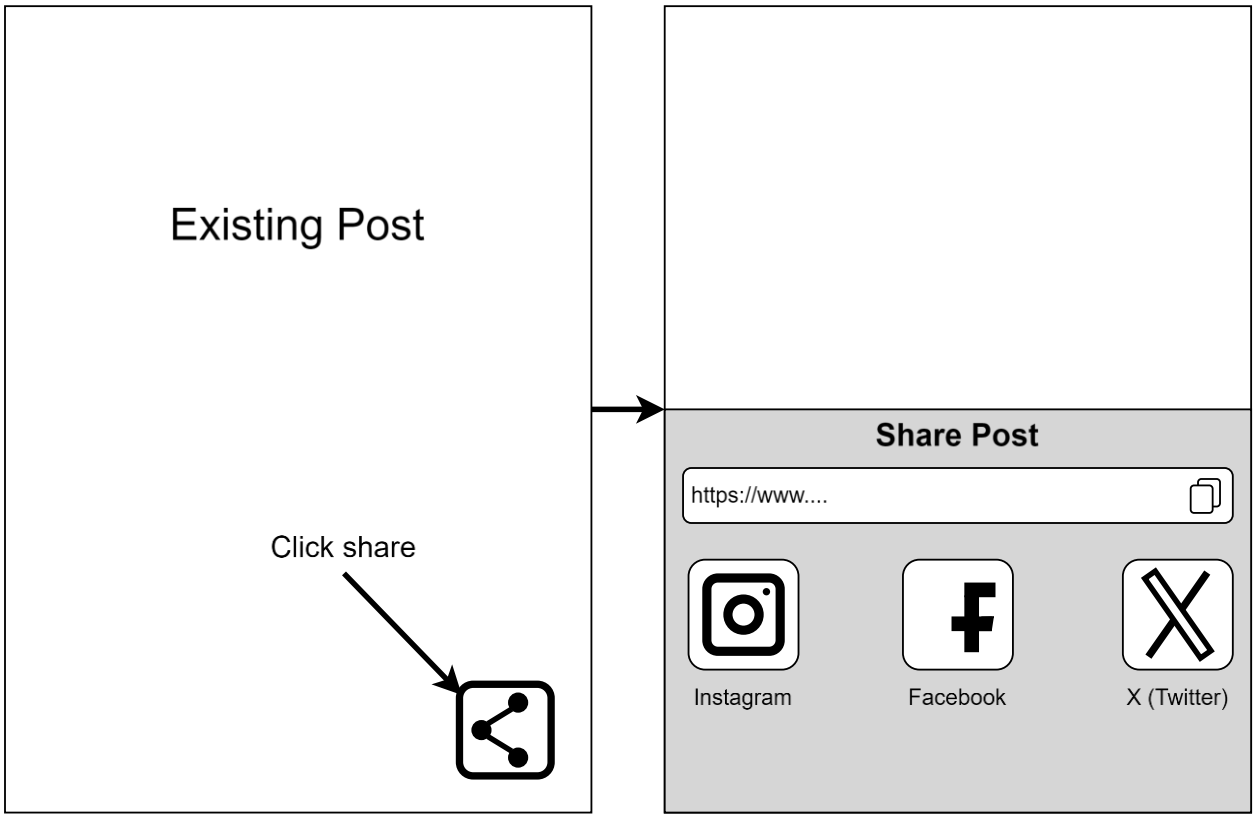
Name	**1.3.1.2 Instagram Post Creator**
Purpose:	Illustrates the component for Instagram post creation
Description:	Illustrates how/where Instagram posts are created in the system and how information is obtained and formatted and how a post is created
Requirements:	EIS.3, EIS.4.1 [PO.1.4.1]
	1.3.1.2.1 Post Info Retriever: fetches the post and user info
	1.3.1.2.2 Post Generator: formats/generates the post
	1.3.1.2.3 Post Sender: sends the generated post to be displayed
	1.1.2 & 3 Uvents Router & Controller
	1.4.1 User Authenticator
	1.3.1.1 Instagram API
Referenced By:	1.3 Social Media Integration
Viewpoint:	Component Diagram

View 1.3.1.4: Get Event Link Data Flow Diagram



Name	1.3.1.4: Get Event Link Data Flow Diagram
Purpose	Display the necessary steps to get a link from the backend to the frontend
Description	The path of the data when a link is requested by a user
Requirements	N/A
Elements	**1.1.1.3.5.3 Events Table**
	(Unknown authenticator view)
	1.3.1.4 Link Generator
Referenced by	1.3
Viewpoint	Data Flow Diagram

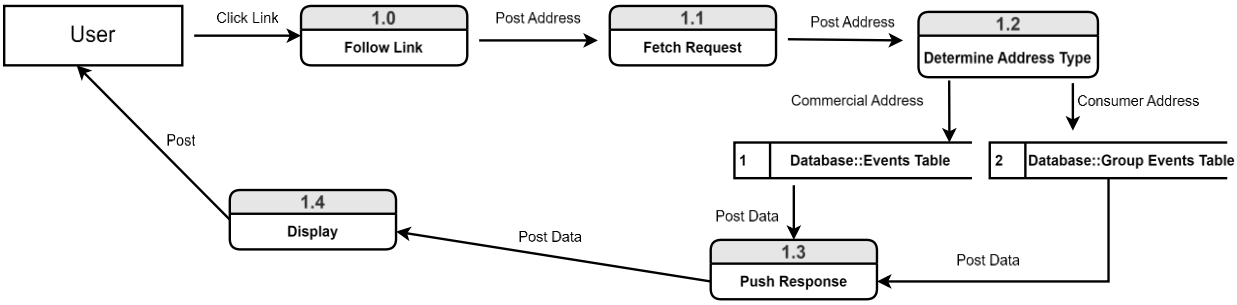
View 1.3.1.5: User Interface for Sharing an Existing Post



Existing Post Share Buttons

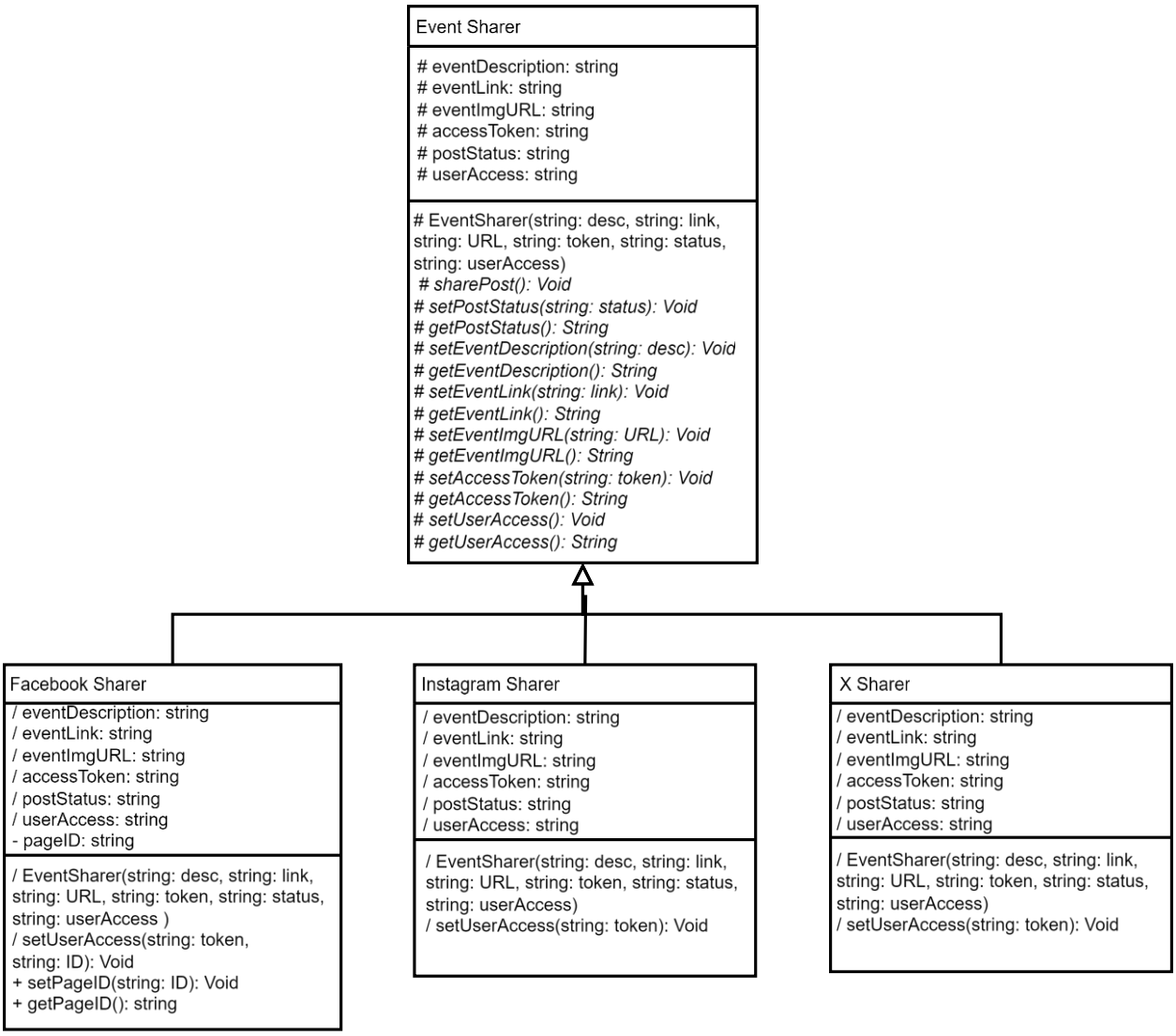
Name	1.3.1.5: User Interface for Sharing an Existing Post
Purpose	The components a user will use to share a link
Description	This view shows the two elements that will be displayed to help share posts. This includes a button and a slide in element with buttons for Instagram, Facebook, and X
Requirements	N/A
Elements	
Referenced by	
Viewpoint	Wire Frame

View 1.3.1.6: Share Link Navigation Data Flow Diagram



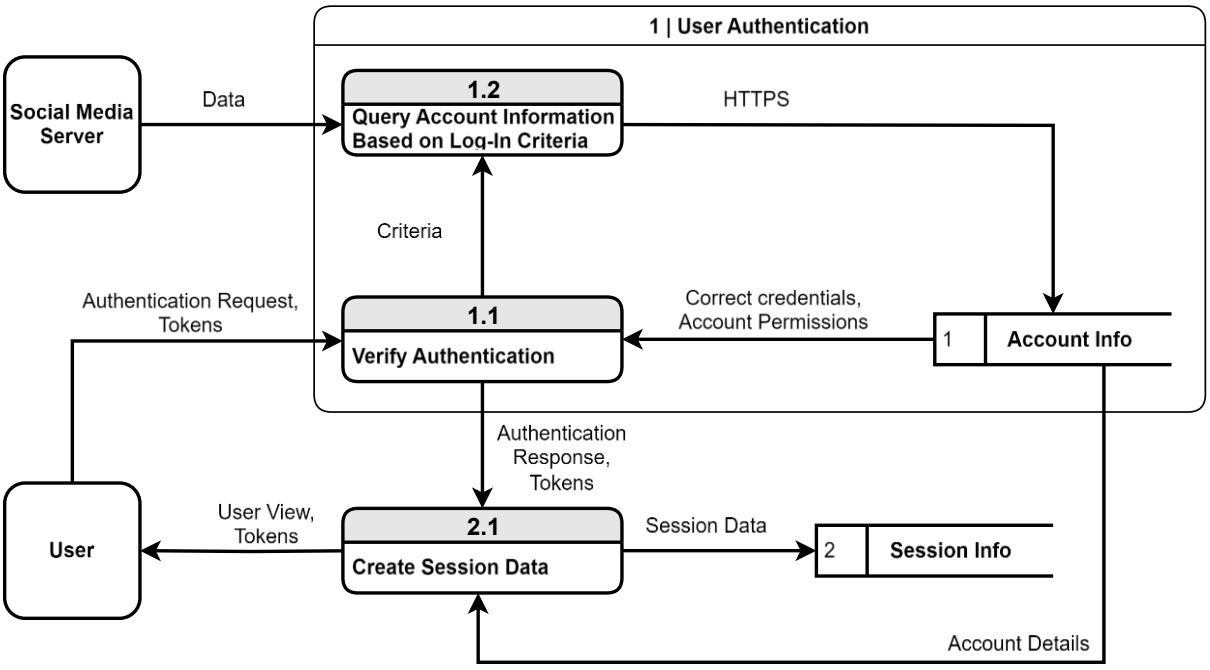
Name	1.3.1.6: Share Link Navigation Data Flow Diagram
Purpose	A view of following a shared link
Description	This view shows what happens when a user clicks on a shared link. The link is sent to the database, which then retrieves the post, and finally the post is displayed to the user.
Requirements	N/A
Elements	**1.3.1.4 Share Link Generator**
	1.1.1.3.5.3 Events Table
	1.1.1.3.5.5 Group Event Table
Referenced by	1.3
Viewpoint	Data Flow Diagram

View 1.3.1.7: Event Sharer Class Diagram



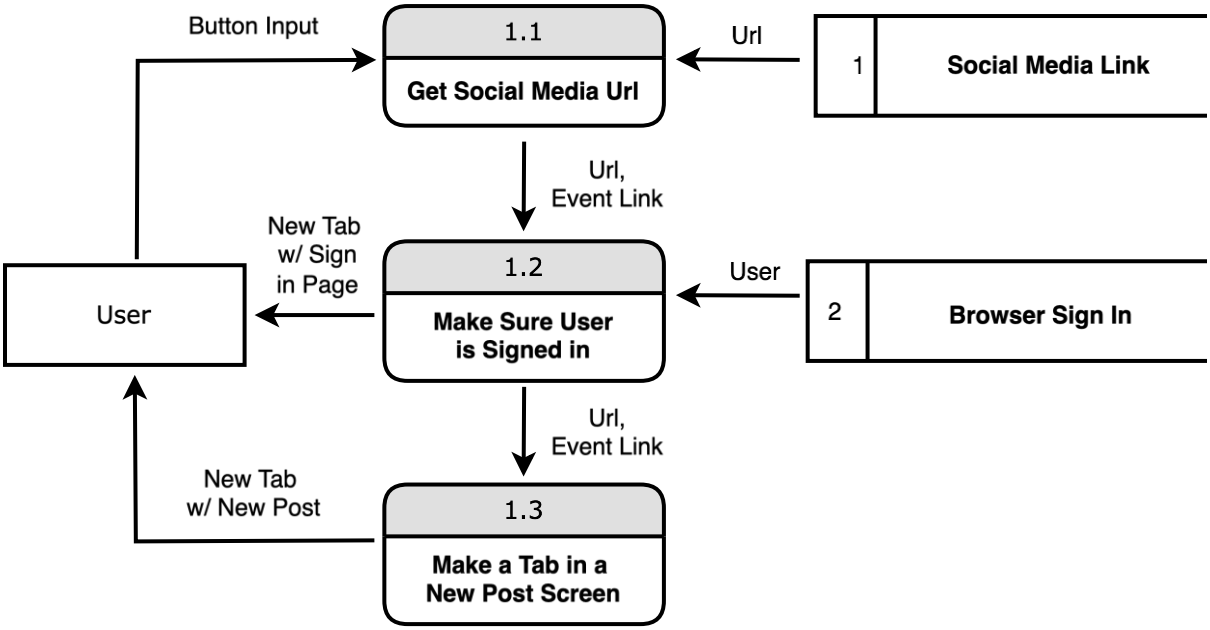
Name	**1.3.1.7 Event Sharer Class Diagram**
Purpose:	Illustrates the functionality and member variables of the Event Sharer base class and that of its children
Description:	Demonstrates how the Event Sharer base class and its children obtain post information, package it, and send it to the social media site to be posted
Requirements:	EIS.3, EIS.4.1 [PO.1.4.1]
Elements	**Event Sharer Base Class**
	Facebook Sharer (child class)
	Instagram Sharer (child class)
	X Sharer (child class)
Referenced By:	1.3.1 Social Media Integration
Viewpoint:	Class Diagram

View 1.3.2: Social Media Account Login



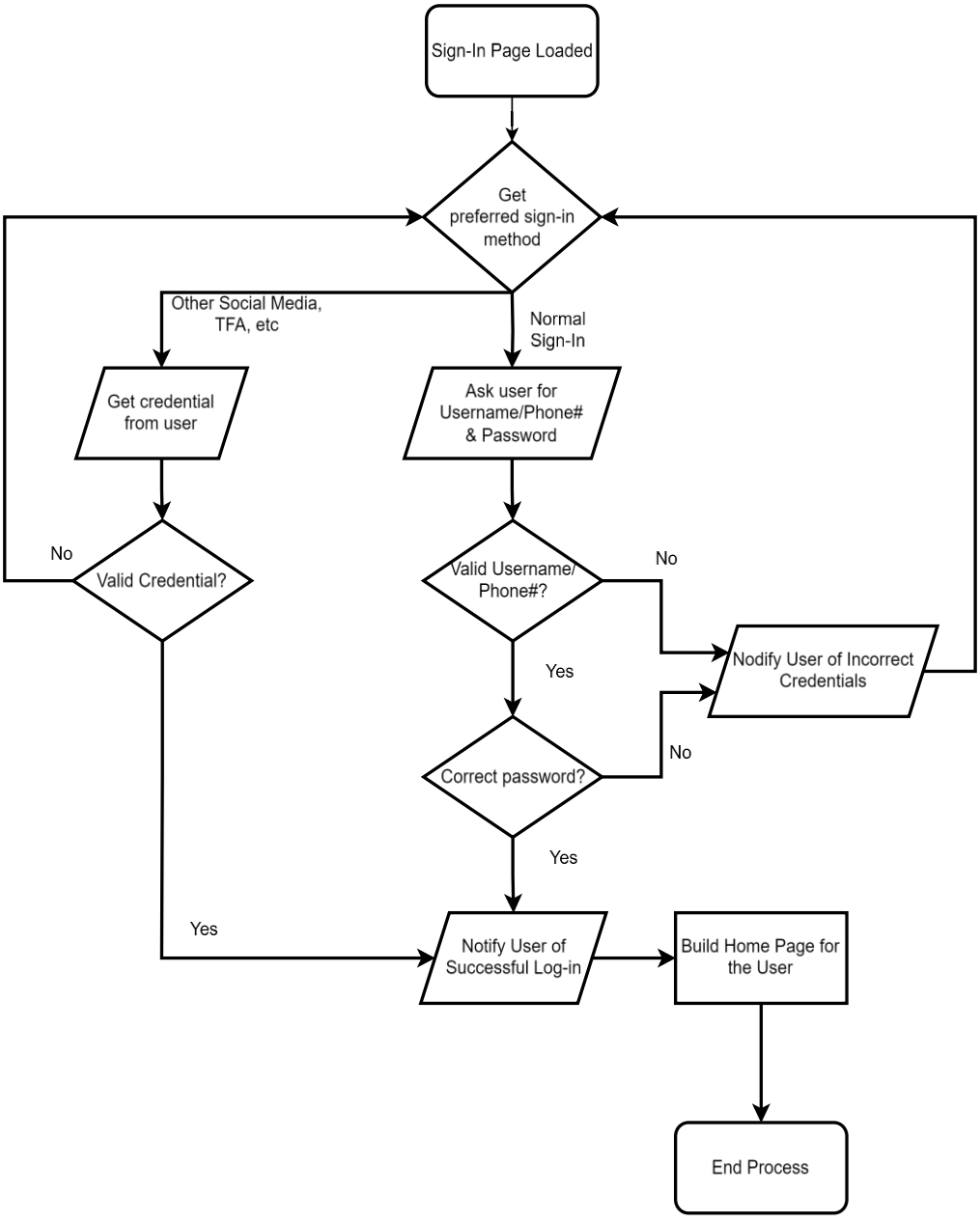
Name	1.3.2: Social Media Account Login
Purpose	Illustrate how data flows during the social media sign-in process.
Description	Illustrates how the database is used to verify the user and generate a view for that user.
Requirements	1.3.1.2, 3.5.1.6
Elements	N/A
Referenced By	1.3.1
Viewpoint	Dataflow Diagram

View 1.3.2.1: Company Social Media Buttons



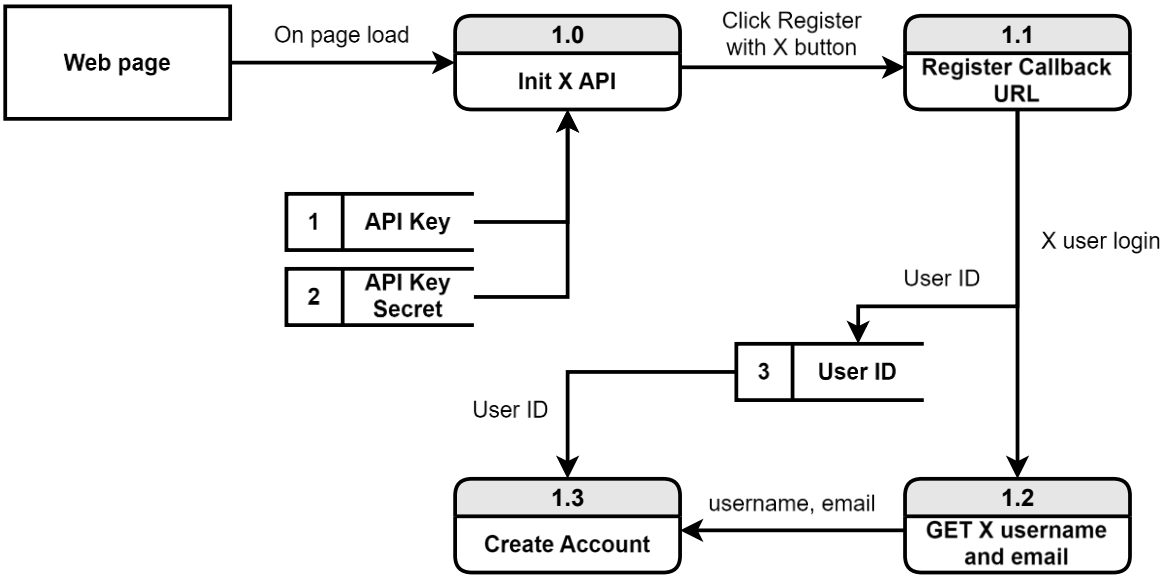
Name	1.3.2.1: Company Social Media Buttons
Purpose	A view of sharing existing events.
Description	This view represents what happens when a user uses a share button while on a company page. It will allow the user to share a link to the event.
Requirements	N/A
Elements	**1.3.1.2: Share Link Generator**
Referenced by	1.3.1
Viewpoint	Data Flow Diagram

View 1.3.2.2: Sign in Flowchart



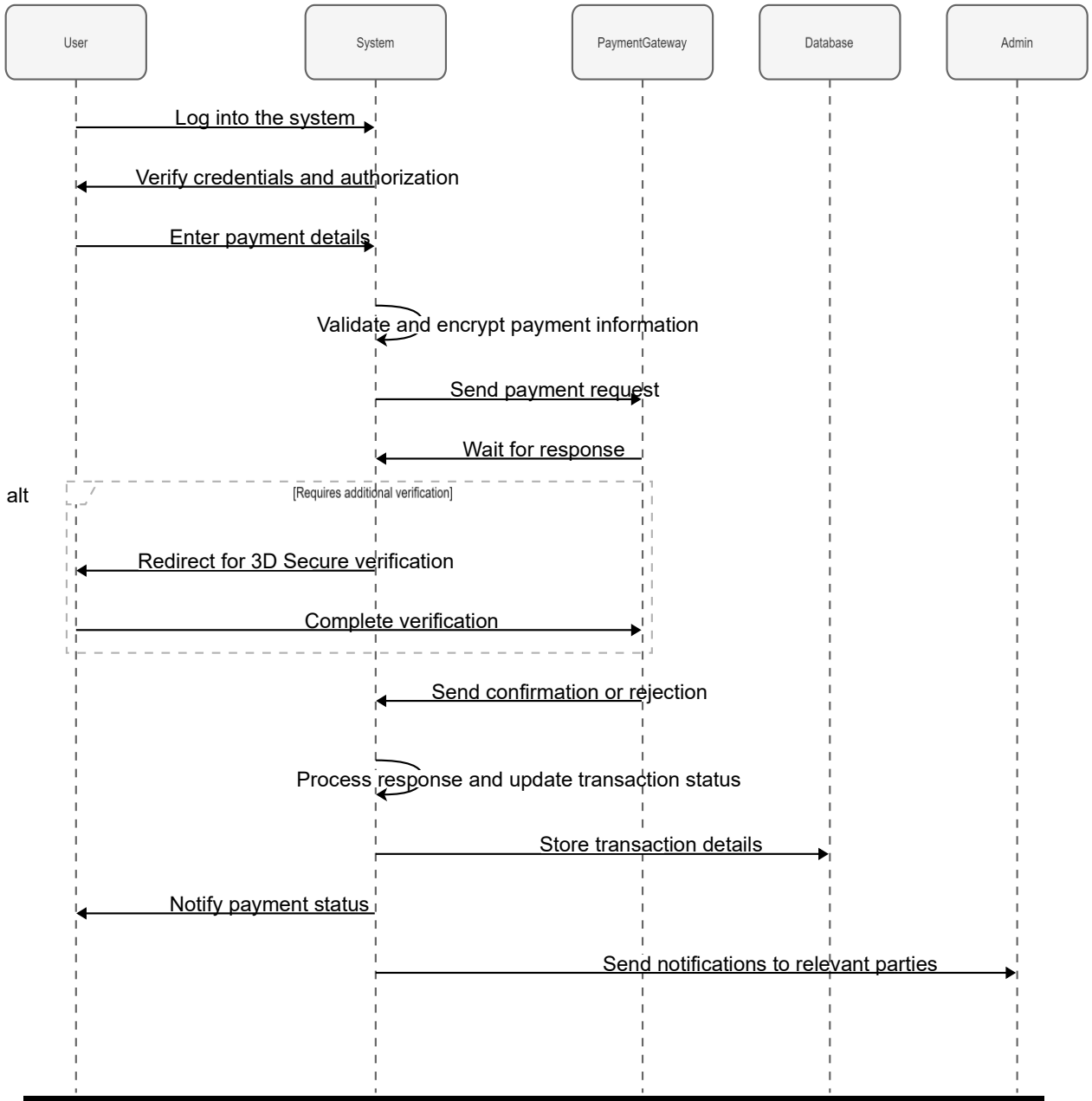
Name	1.3.2.2: Sign in Flowchart
Purpose	Illustrate the decisions that are made during the sign-in process.
Description	Illustrates how the social media sites verify if the login criteria are correct.
Requirements	1.3.1.2, 3.5.1.6
Elements	N/A
Referenced By	1.3
Viewpoint	Flowchart

View 1.3.3.1 Create Account X



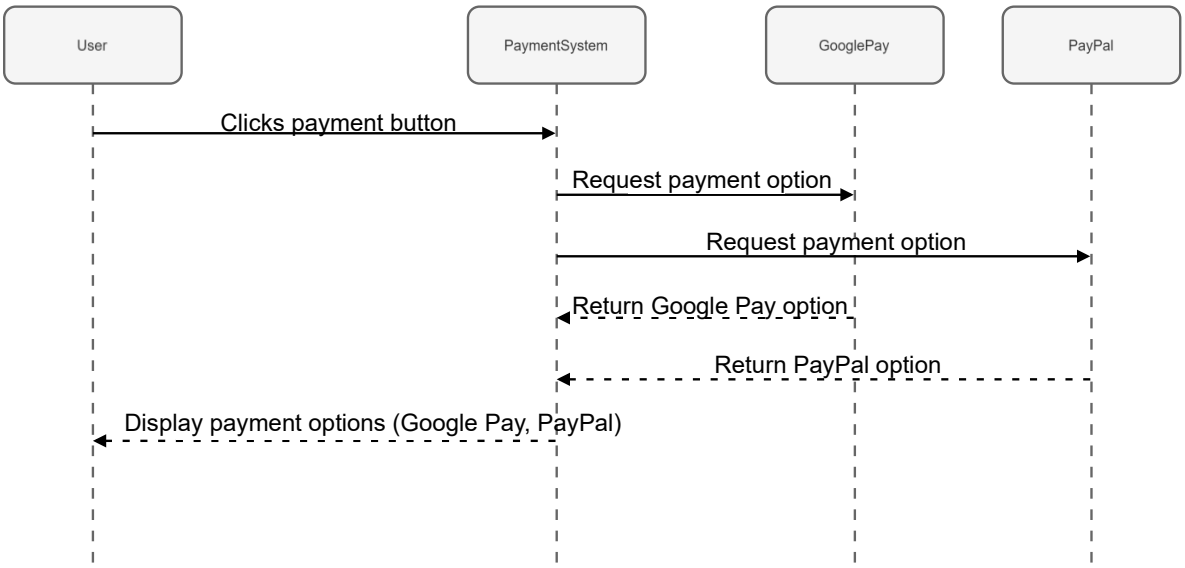
Name	**1.3.3.1 Create Account X**
Purpose:	Illustrate the flow of creating an account with an existing X account.
Description:	This view illustrates the steps taken when creating an account with an existing X account.
Requirements:	EIS.4
Elements:	**Web page**
	1.3.3.1 Init X API: Requires API Key and API Key Secret
	1.3.3.1.1 Register Callback URL: Returns User ID after X user login
	1.3.3.2 GET X username and email
	1.3.3.1.3 Create Account
Referenced By:	1.3 X Account Creator
Viewpoint:	Data Flow Diagram

1.4 Payment System Overview



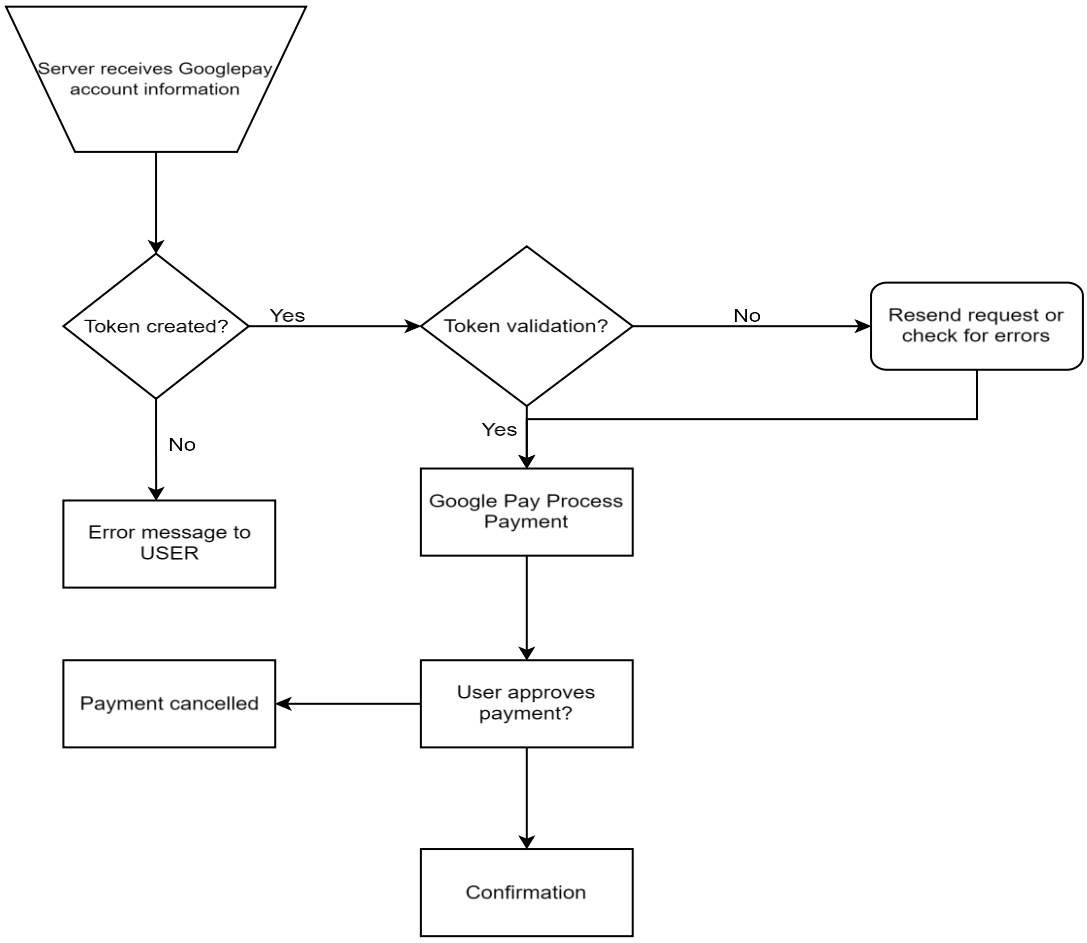
Name	View 1.4 Payment System Overview
Description	Overview of the payment system
Purpose	The program automatically displays the order summary once the user signs in with PayPal
Requirements	EIS.6, FS.6, PO.2.1.6
Elements	
Referenced By	1.4 Payment
Viewpoint	Workflow Diagram

View 1.4.1: Return Payment Options



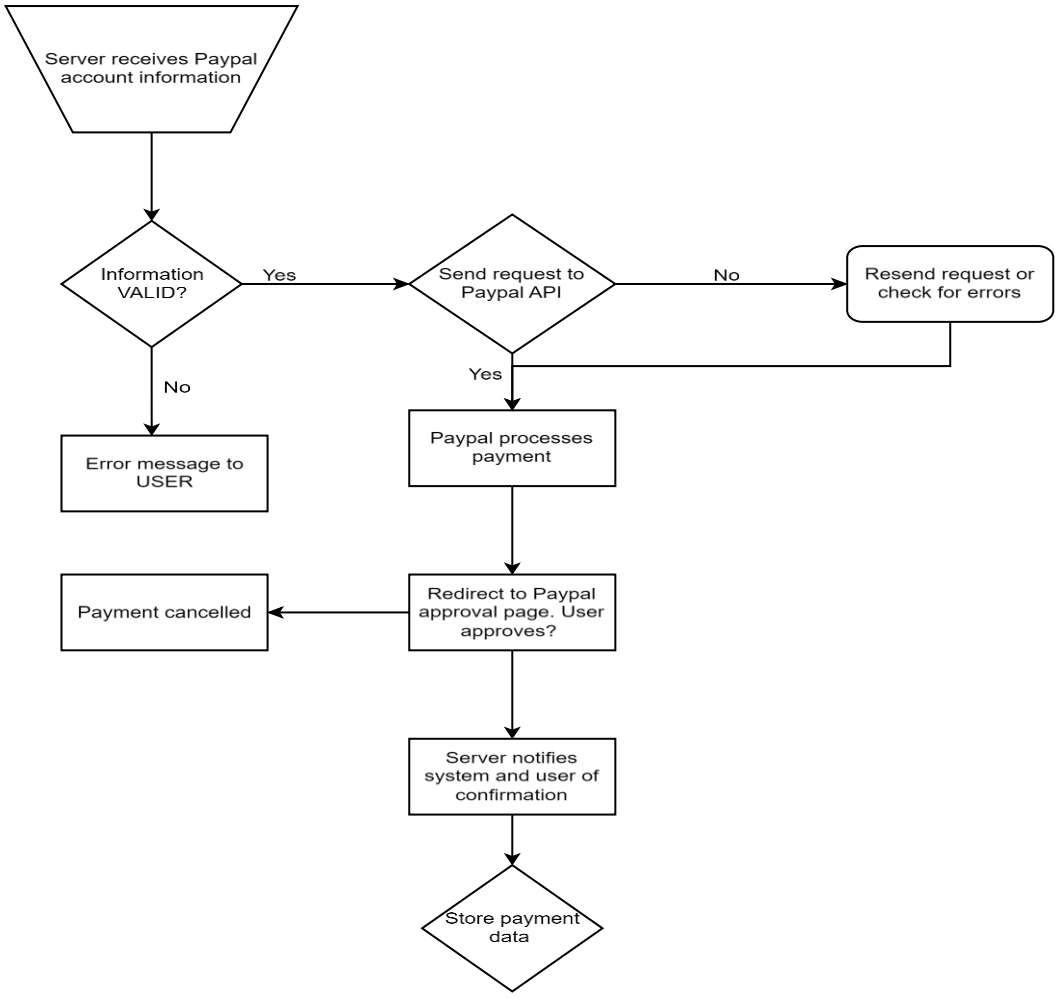
Name	View 1.4.1 Return Payment Options
Purpose	To render Google Pay and PayPal functions to the user.
Description	This view will show that Google Pay and PayPal will be returned as options once the user clicks the payment button.
Requirements	Backend system capable of handling requests to Google Pay and PayPal, the user interface to display payment options.
Elements	**User** **Payment System** **Google Pay** **PayPal**
Referenced By	Sequence diagram illustrating the interaction between user and backend payment functions.
Viewpoint	Workflow diagram.

View 1.4.1.1: Google Pay



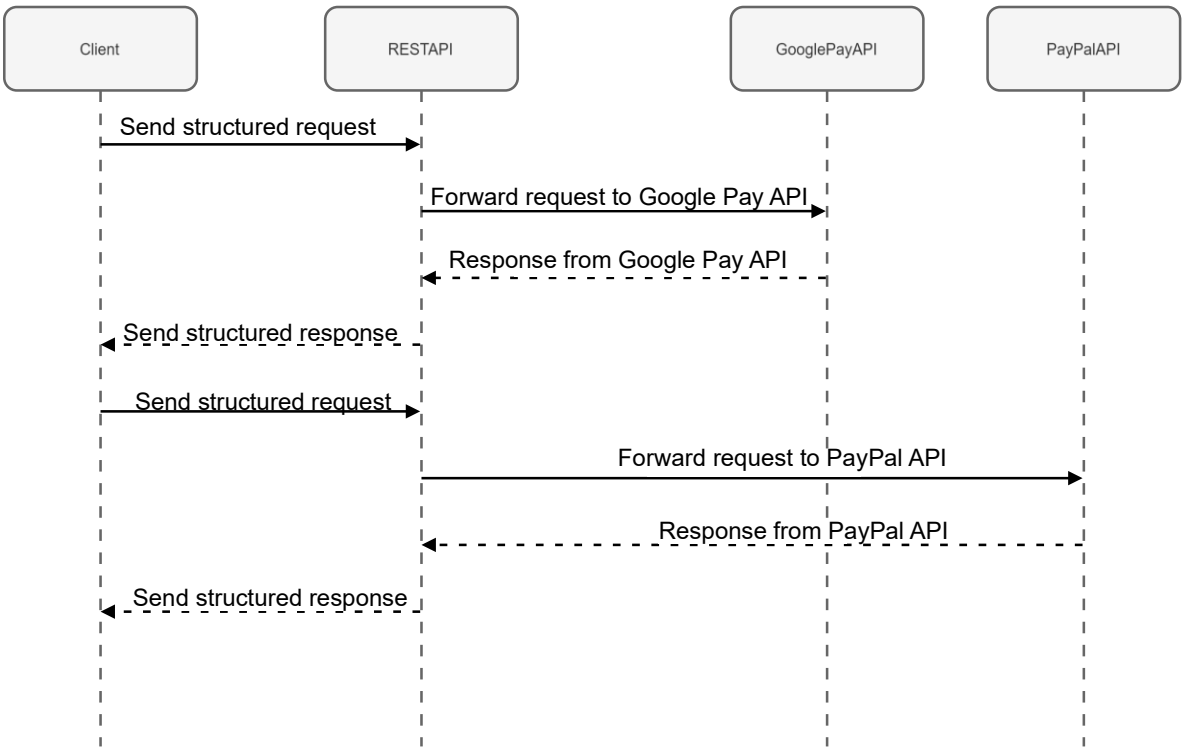
Name	View 1.4.1.1
Purpose	Describe Google Pay payment process
Description	Flow diagram the shows the movement of data at the different payment processing steps when using Google Pay
Requirements	Backend system capable of retrieving and displaying order summary, user interface to display order summary, integration with Google Pay for user authentication.
Elements	**1.4.1.1.1 Order Summary** **1.4.1.1.2 Payment Confirmation** **1.4.1.1.3 Error Handling**
Referenced By	View 1.4.1 Payment Method Selection
Viewpoint	Workflow Diagram

View 1.4.1.2: PayPal



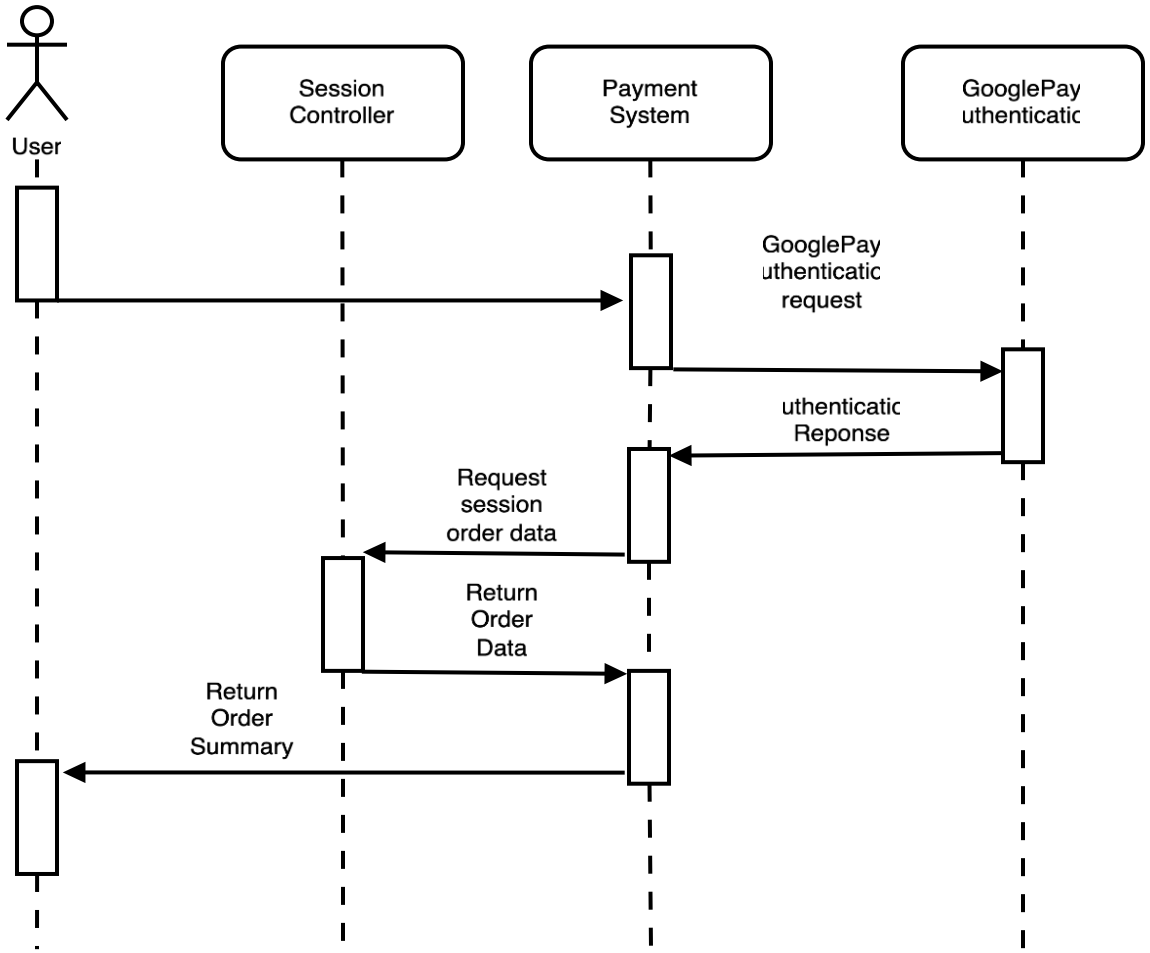
Name	View 1.4.1.2
Purpose	Describe PayPal payment process
Description	Flow diagram the shows the movement of data at the different payment processing steps when using PayPal
Requirements	Backend system capable of retrieving and displaying order summary, user interface to display order summary, integration with PayPal for user authentication.
Elements	**1.4.1.2.1 Payment Confirmation** **1.4.1.2.2 Error Handling** **1.4.1.2.3 Order Summary**
Referenced By	View 1.4.1 Payment Method Selection
Viewpoint	Workflow Diagram

View 1.4.1.5: Implement REST API



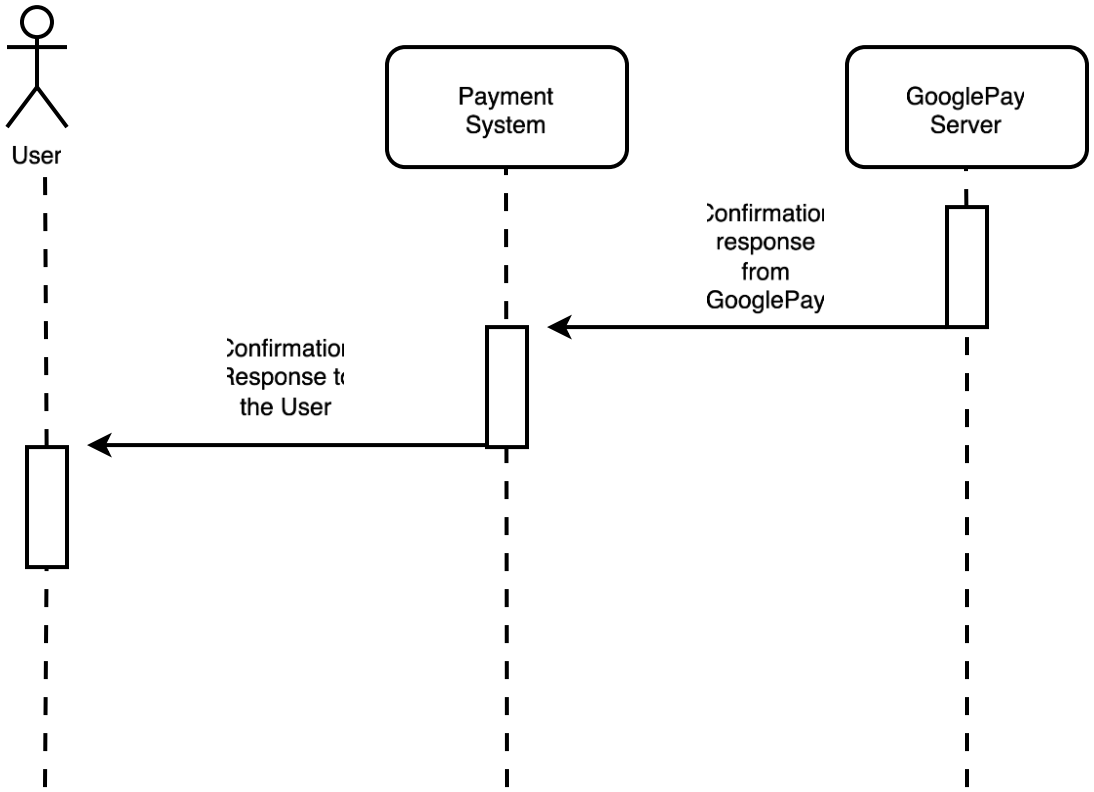
Name	**View 1.4.1.5: Implement REST API.**
Description	A REST API to facilitate structured requests and responses between the client and the Google Pay and PayPal APIs.
Purpose	Implement REST API for structured requests and responses with Google Pay and PayPal APIs.
Requirements	A system capable of handling REST API requests, forwarding them to the appropriate payment APIs, and structuring the responses.
Elements	**Client,** **REST API,** **GooglePayAPI,** **PayPalAPI**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.1.1: Google Pay Order Summary



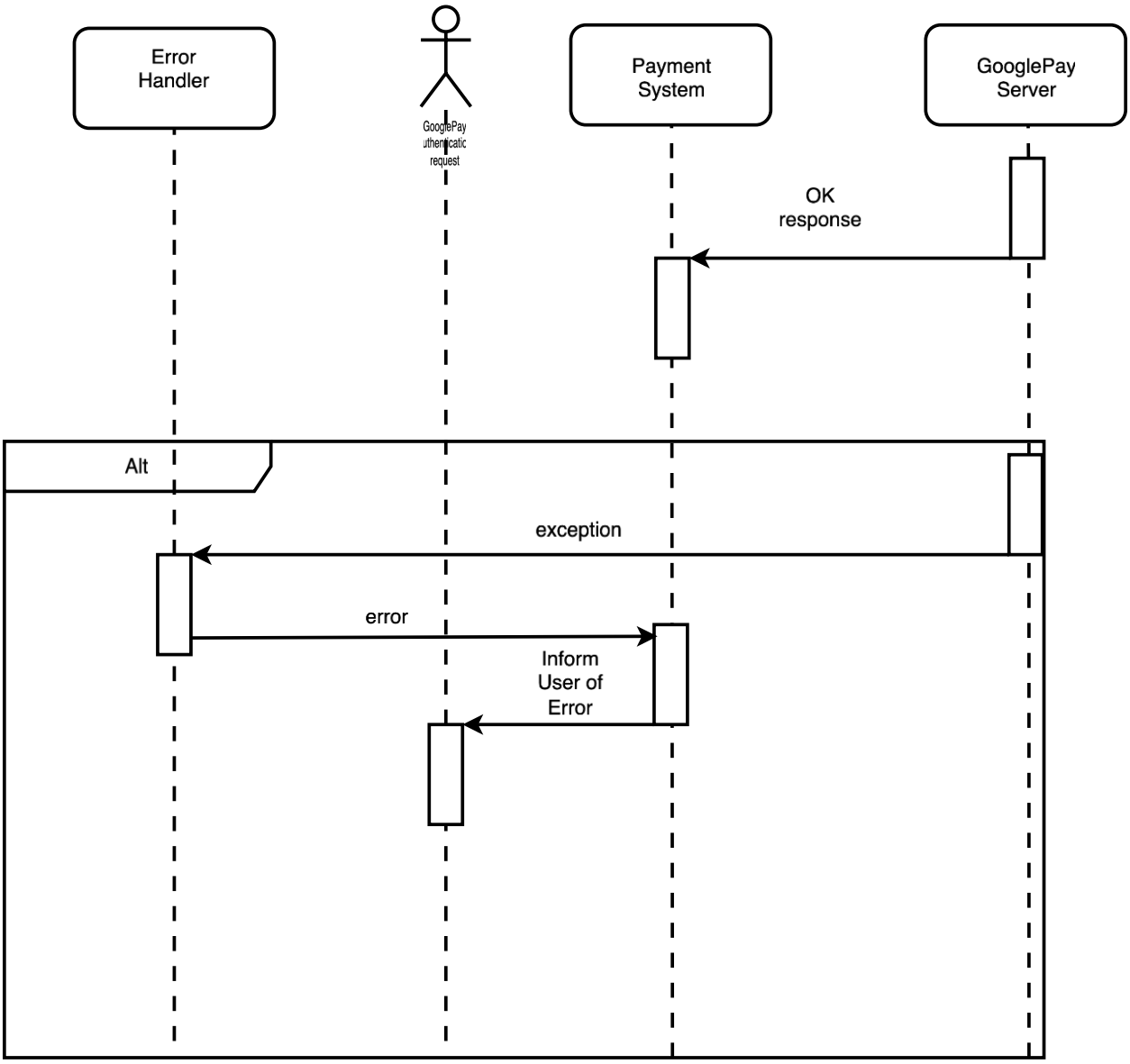
Name	View 1.4.1.1.1: Google Pay Order Summary
Description	The application automatically displays the order summary once the user signs in with Google Pay
Purpose	The application automatically displays the order summary once the user signs in with Google Pay
Requirements	EIS.6, FS.6, PO.2.1.6
Elements	**User** **Session Controller** **Payment System**
Referenced By	1.4 Payment
Viewpoint	Sequence Diagram

View 1.4.1.1.2: Google Pay Error Handling (Confirmation)



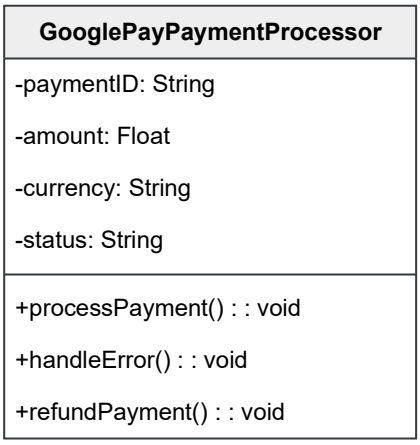
Name	View 1.4.1.1.2: Google Pay Payment Confirmation
Description	If anything goes wrong within the payment process, it should be handled.
Purpose	Display payment confirmation to the user.
Requirements	EIS.6, FS.6, PO.2.1.6
Elements	**Payment System** **User**
Referenced By	1.4 Payment
Viewpoint	Sequence Diagram

View 1.4.1.1.3: Google Pay Error Handling



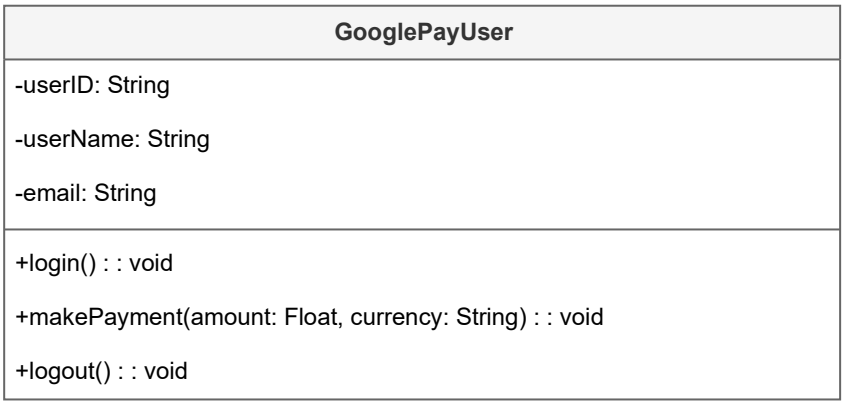
Name	View 1.4.1.1.3: Google Pay Error Handling
Description	If anything goes wrong within the payment process, it should be handled.
Purpose	Handles Google Pay Errors if they arise
Requirements	EIS.6, FS.6, PO.2.1.6
Elements	**Payment System** **User** **Error Handler**
Referenced By	1.4 Payment
Viewpoint	Sequence Diagram

View 1.4.1.1.4: Google Pay Payment Processor Class



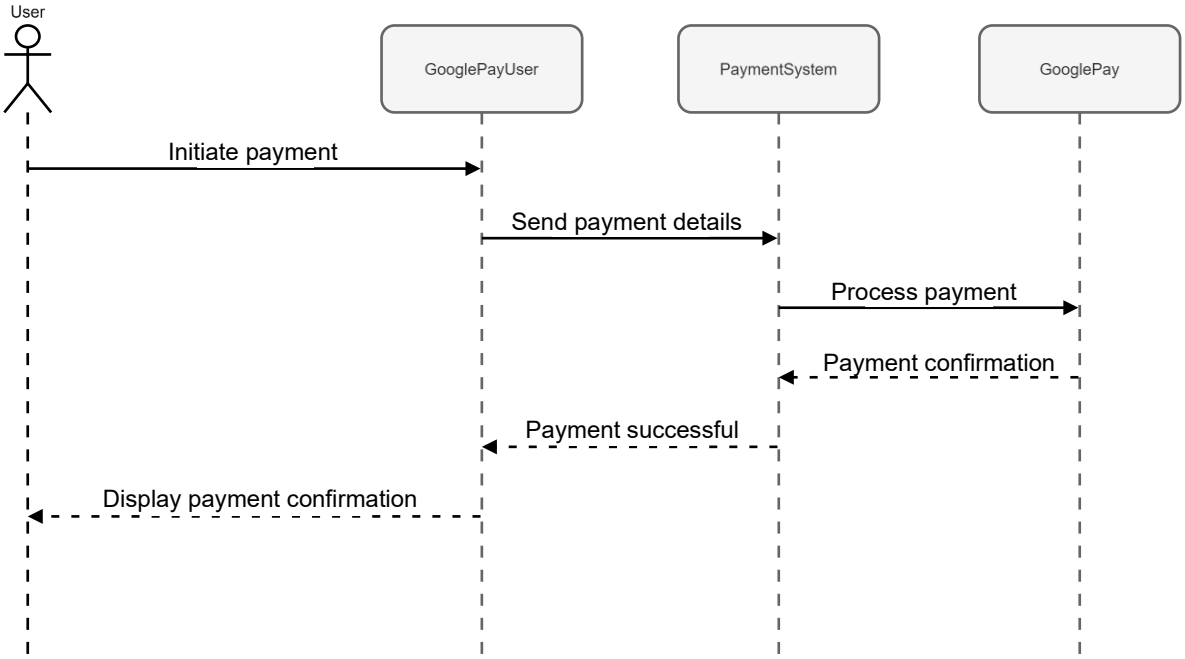
Name	View 1.4.1.1.4: GooglePay Payment Processor Class
Description	Handles the payment processing logic.
Purpose	Handles the payment processing logic.
Requirements	Class capable of processing payments, handling errors, and refunding payments.
Elements	**GooglePayPaymentProcessor**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.1.4.1: Google Pay User Class



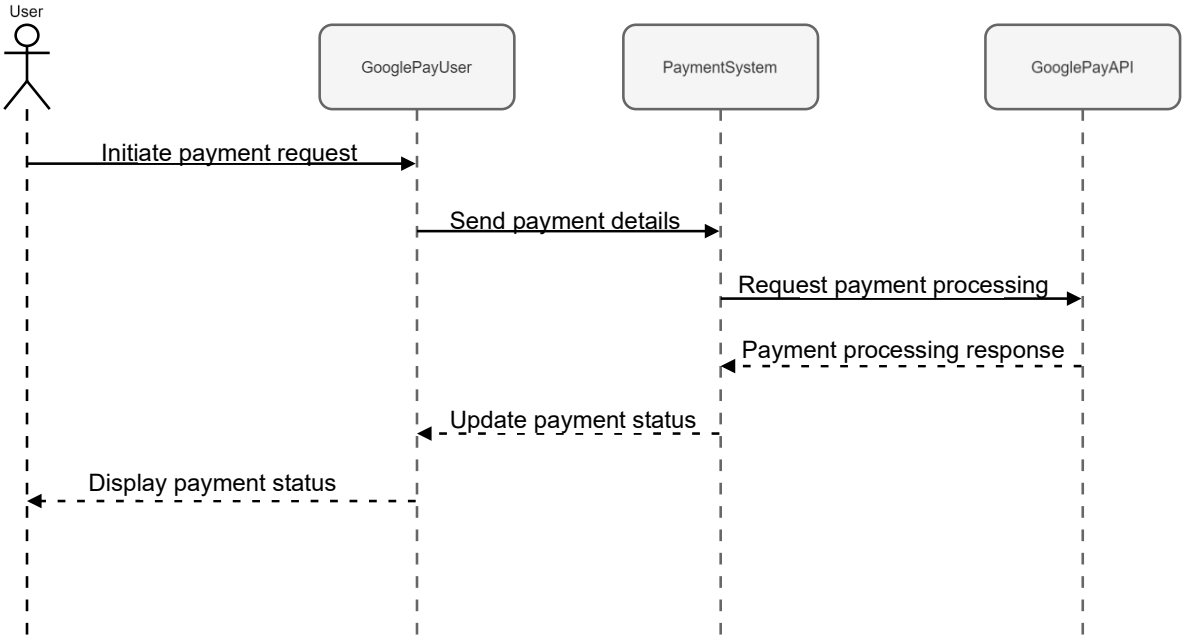
Name	View 1.4.1.1.4.1: Google Pay User Class
Description	Shows a user making a Google Pay payment.
Purpose	Shows a user making a Google Pay payment.
Requirements	Class capable of representing a user, handling login, payment, and logout actions.
Elements	**GooglePayUser**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.1.4.2: Google Pay Transaction



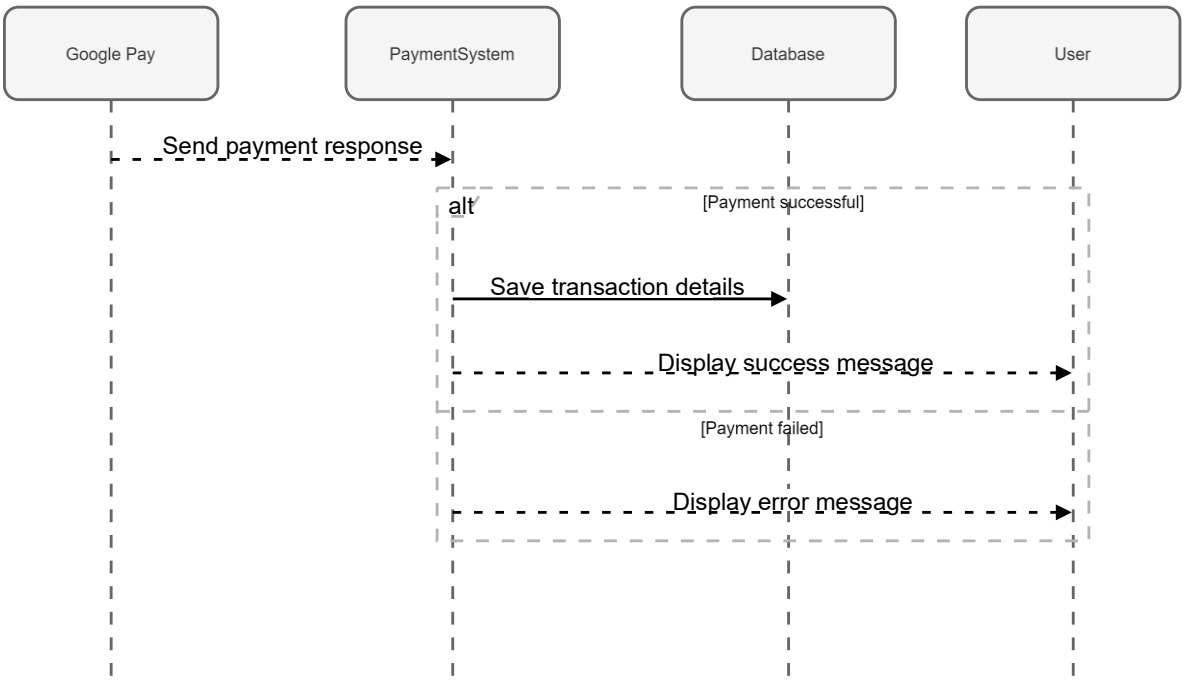
Name	View 1.4.1.1.4.2: Google Pay Transaction
Description	Shows a Google Pay payment transaction.
Purpose	Shows a Google Pay payment transaction.
Requirements	System capable of handling PayPal payment transactions, user interface to initiate and confirm transactions.
Elements	**User,** **GooglePayUser,** **PaymentSystem,** **GooglePay**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.1.5: Google Pay Payment Request



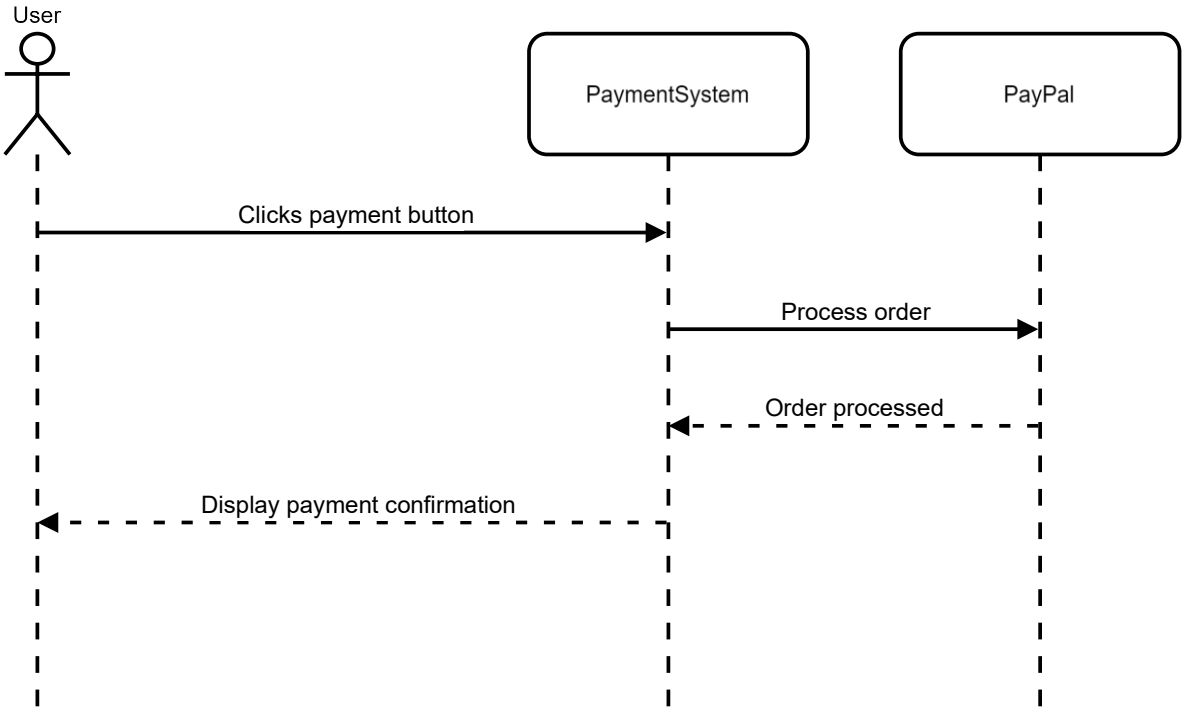
Name	View 1.4.1.1.5: Google Pay Payment Request
Description	Handles Google Pay payment requests to the Google Pay API.
Purpose	Manages the process of sending payment requests to the Google Pay API and handling the responses.
Requirements	A system capable of sending payment requests to Google Pay API, handling responses, and updating payment status.
Elements	**User,** **GooglePayUser,** **PaymentSystem,** **GooglePayAPI**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.1.5.1: Google Pay Payment Response



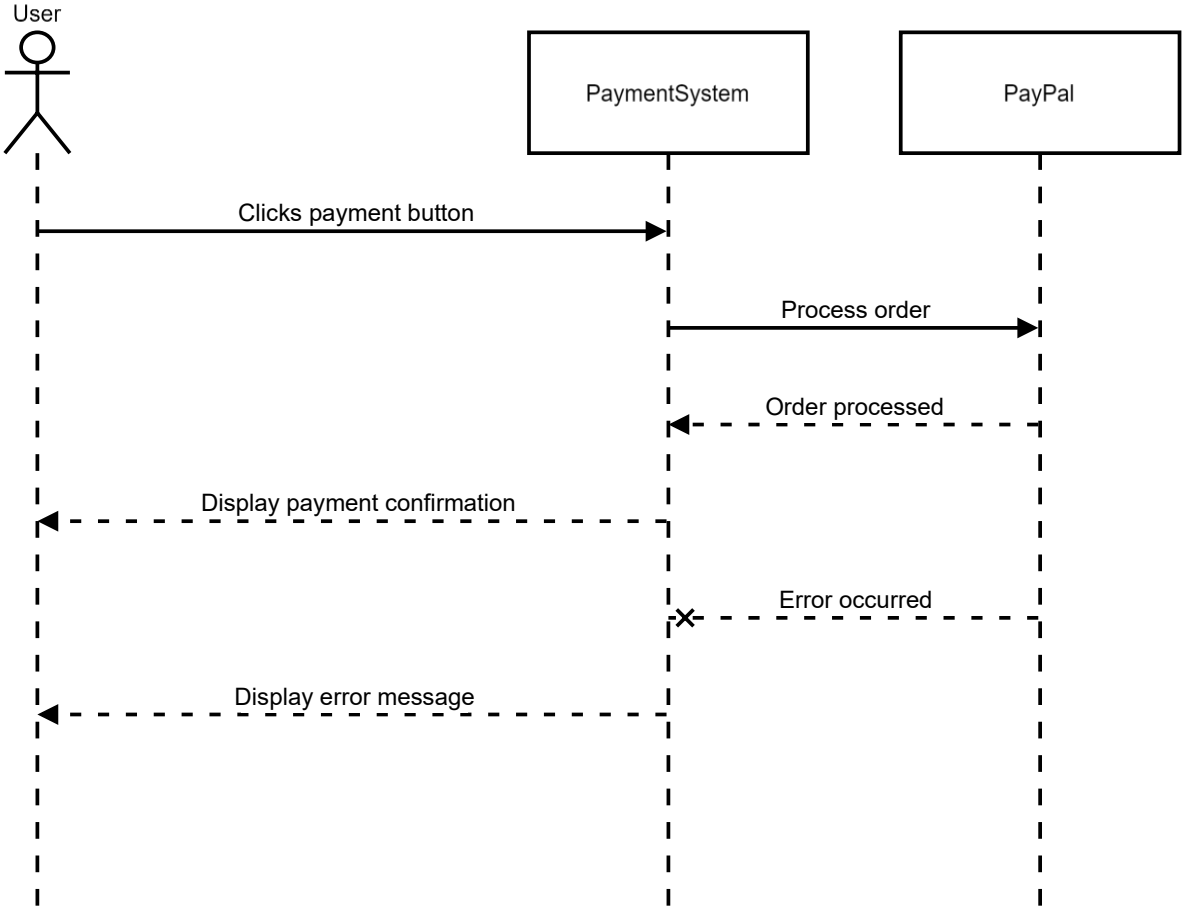
Name	View 1.4.1.1.5: Google Pay Payment Response
Description	Handles responses from the Google Pay API.
Purpose	Processes and manages the responses received from the Google Pay API to ensure proper handling of successful and failed transactions.
Requirements	A system capable of handling Google Pay API responses, storing transaction details, and updating user interface based on response.
Elements	**GooglePay,** **PaymentSystem** **Database,** **User**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.2.1: PayPal Payment Confirmation



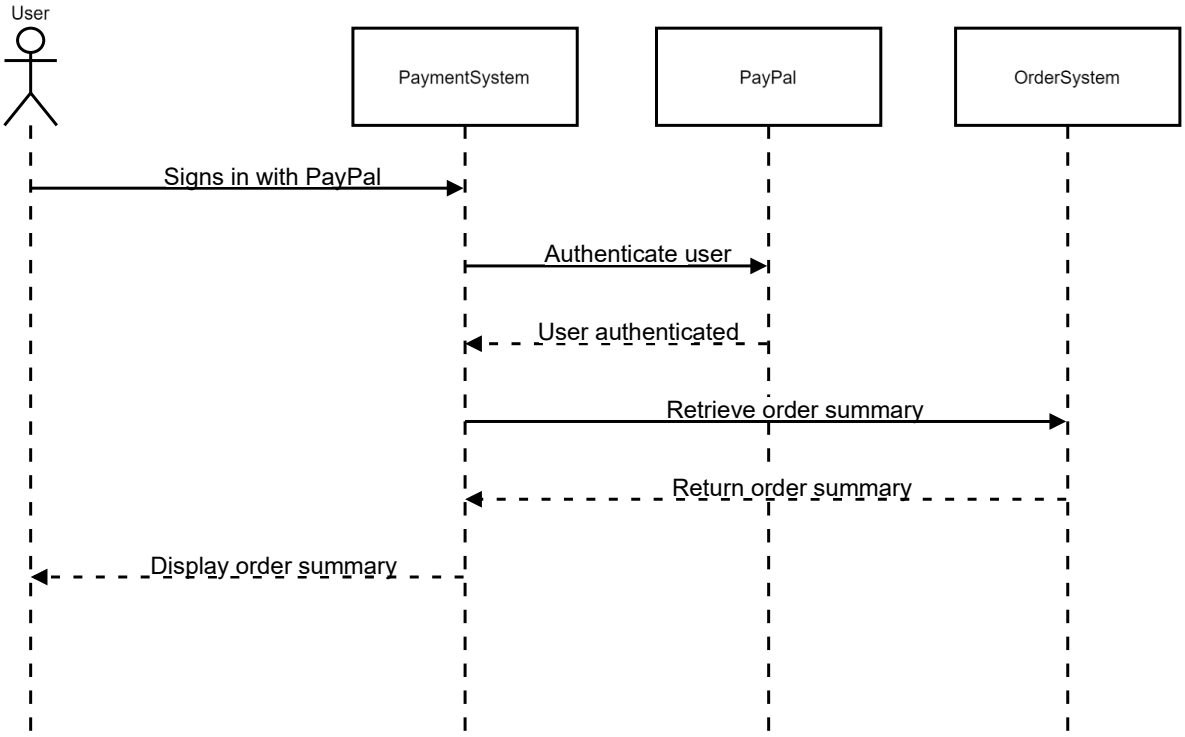
Name	View 1.4.1.2.1: PayPal Payment Confirmation
Purpose	Display payment confirmation to the user.
Description	Once PayPal API completes the order, the user should see the payment confirmation.
Requirements	Backend system capable of handling requests and responses from PayPal, the user interface to display payment confirmation.
Elements	**User** **Payment System**
Referenced By	1.4.2 Payment Options
Viewpoint	Sequence Diagram

View 1.4.1.2.2: PayPal Error Handling



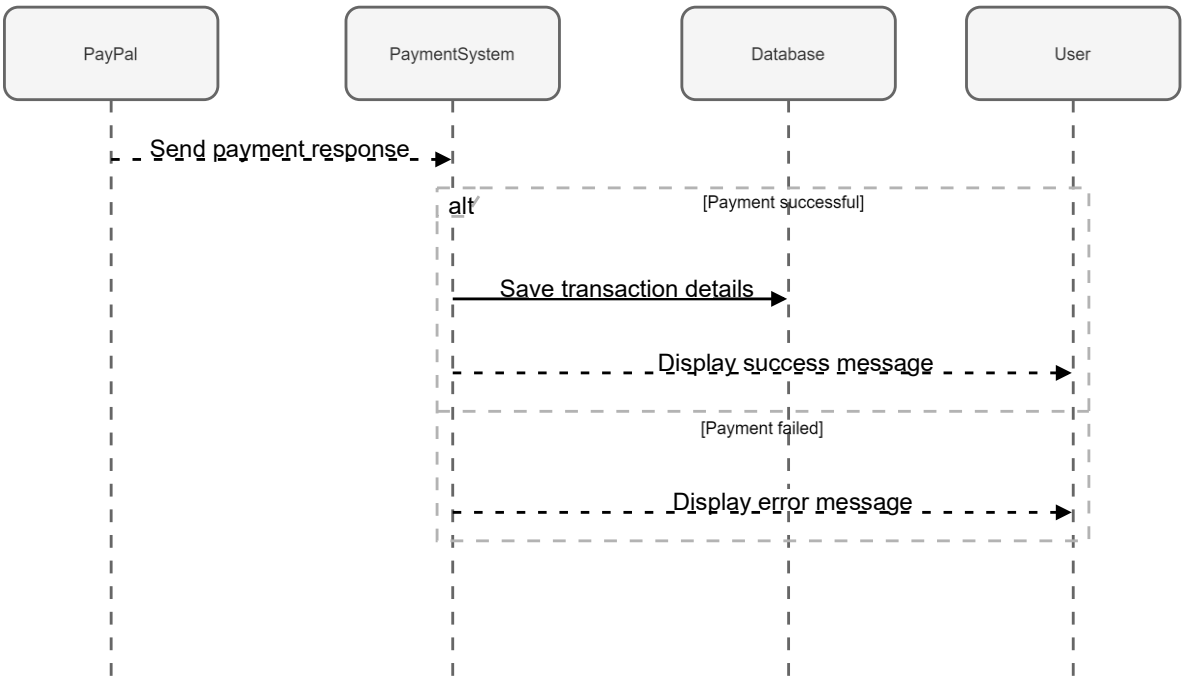
Name	View 1.4.1.2.2 PayPal Error Handling
Purpose	Provide PayPal error handling if any errors arise.
Description	It should handle any error that could potentially arise from PayPal API
Requirements	A backend system capable of detecting and handling errors from PayPal, user interface to display error messages.
Elements	**User** **Payment System**
Referenced by	1.4.2 Payment Options
Viewpoint	Sequence Diagram

View 1.4.1.2.3: PayPal Order Summary



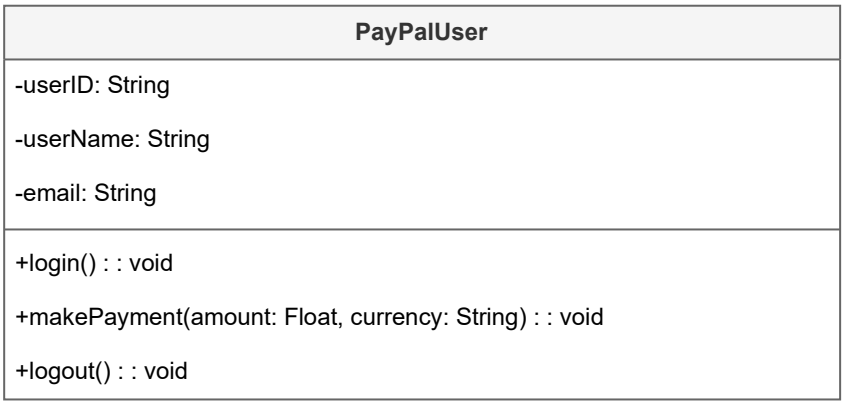
Name	View 1.4.1.2.3: PayPal Order Summary
Description	Load the user order summary
Purpose	The program automatically displays the order summary once the user signs in with PayPal
Requirements	Backend system capable of retrieving and displaying order summary, user interface to display order summary, integration with PayPal for user authentication.
Elements	**User** **PaymentSystem** **PayPal** **OrderSystem**
Referenced By	1.4 Payment
Viewpoint	Sequence Diagram

View 1.4.1.2.4: PayPal Payment Response



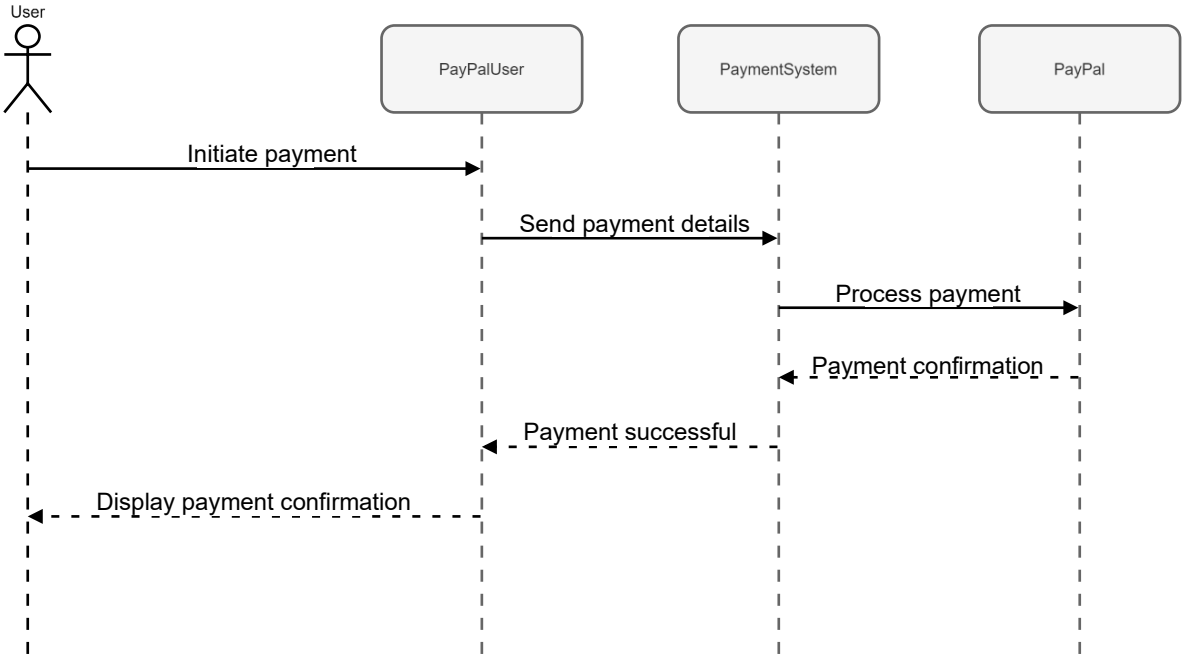
Name	**View 1.4.1.2.4:** PayPal Payment Response
Description	Handles responses from the PayPal API.
Purpose	Processes and manages the responses received from the PayPal API to ensure proper handling of successful and failed transactions.
Requirements	A system capable of handling PayPal API responses, storing transaction details, and updating user interface based on response.
Elements	**PayPal,** **PaymentSystem** **Database,** **User**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.2.4.1: PayPal User Class



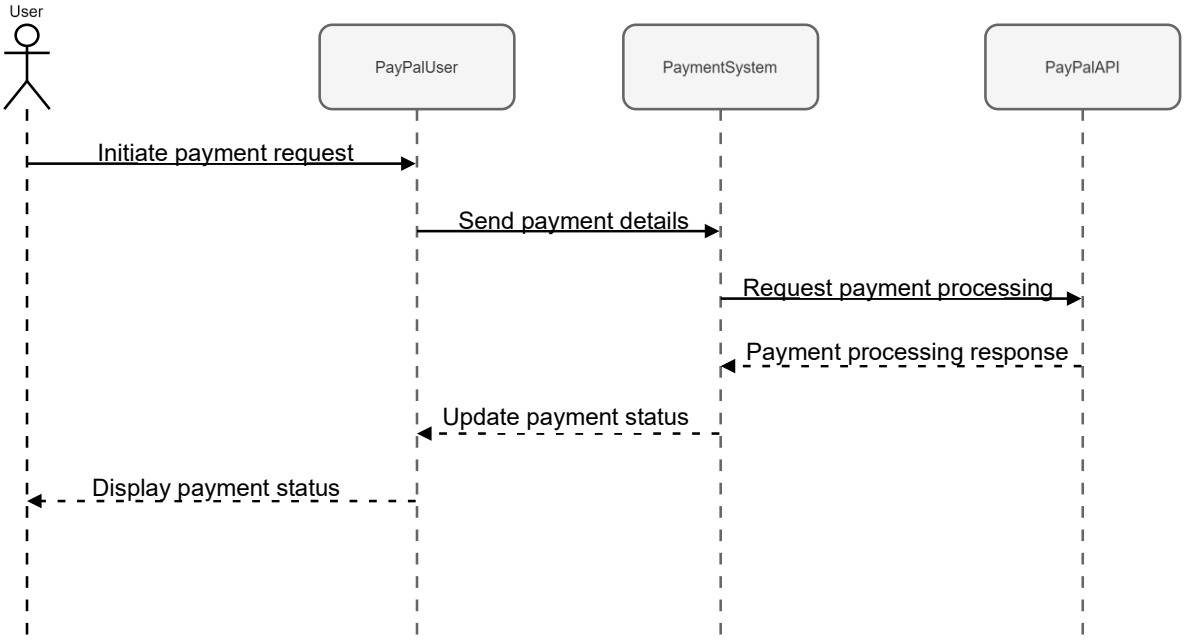
Name	View 1.4.1.2.4.1: PayPal User Class
Description	Shows a user making a PayPal payment.
Purpose	Shows a user making a PayPal payment.
Requirements	Class capable of representing a user, handling login, payment, and logout actions.
Elements	**PayPalUser**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.2.4.2: PayPal Transaction



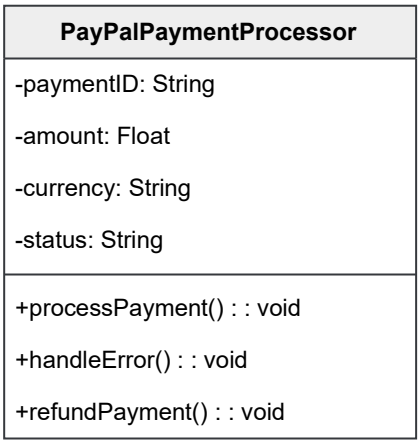
Name	View 1.4.1.2.4.2: PayPal Transaction
Description	Shows a PayPal payment transaction.
Purpose	Shows a PayPal payment transaction.
Requirements	System capable of handling PayPal payment transactions, user interface to initiate and confirm transactions.
Elements	**User,** **PayPalUser,** **PaymentSystem,** **PayPal**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.2.5: PayPal Payment Request



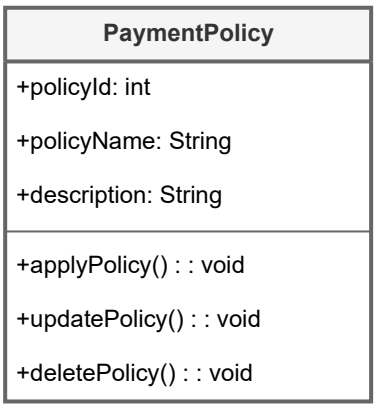
Name	View 1.4.1.2.5: PayPal Payment Request
Description	Handles PayPal Pay payment requests to the PayPal API.
Purpose	Manages the process of sending payment requests to the PayPal API and handling the responses.
Requirements	A system capable of sending payment requests to PayPal API, handling responses, and updating payment status.
Elements	**User,** **PayPalUser,** **PaymentSystem,** **PayPalAPI**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.2.6: PayPal Payment Processor Class



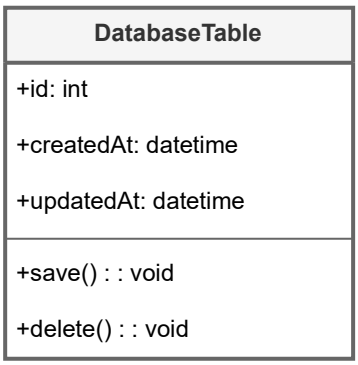
Name	View 1.4.1.2.6: PayPal Payment Processor Class
Description	Handles the payment processing logic.
Purpose	Handles the payment processing logic.
Requirements	Class capable of processing payments, handling errors, and refunding payments.
Elements	**PayPalPaymentProcessor**
Referenced By	
Viewpoint	Class Diagram

View 1.4.2: Payment Policy



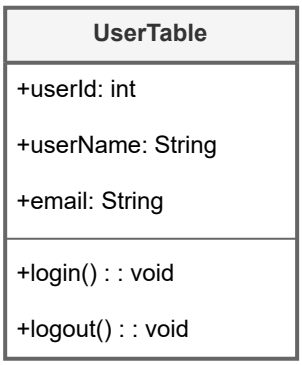
Name	View 1.4.2: Payment Policy
Description	Defines and applies various payment policies to ensure consistent payment processing.
Purpose	Contains and manages all the payment policies for the system.
Requirements	System capable of storing, updating, and deleting payment policies, and applying them during transactions.
Elements	**PaymentPolicy**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.3: Database Table



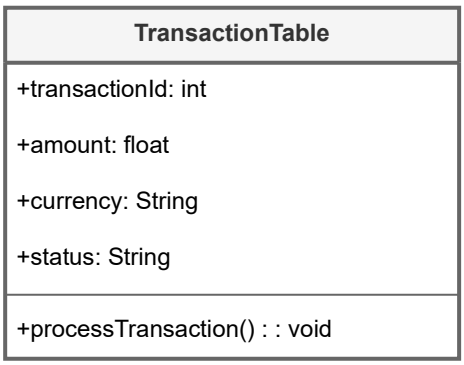
Name	**View 1.4.1.3**: Database Table
Description	Serves as the parent class for the user, transaction, and payment tables.
Purpose	Provides a common structure and methods for handling database operations for user, transaction, and payment tables.
Requirements	A system capable of managing database tables with common functionality and ensuring data consistency.
Elements	**DatabaseTable**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.3.1: User Table



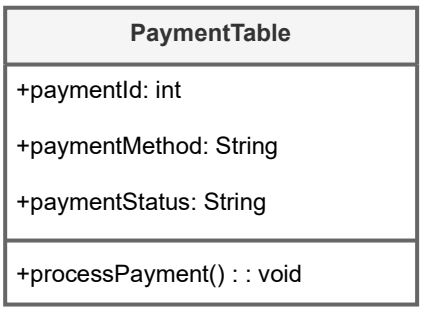
Name	**View 1.4.1.3.1: User Table**
Description	Stores the user information and manages user authentication.
Purpose	Contains user details and provides methods for user login and logout.
Requirements	Database table for storing user information and handling the authentication process.
Elements	**UserTable**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.3.2: Transaction Table



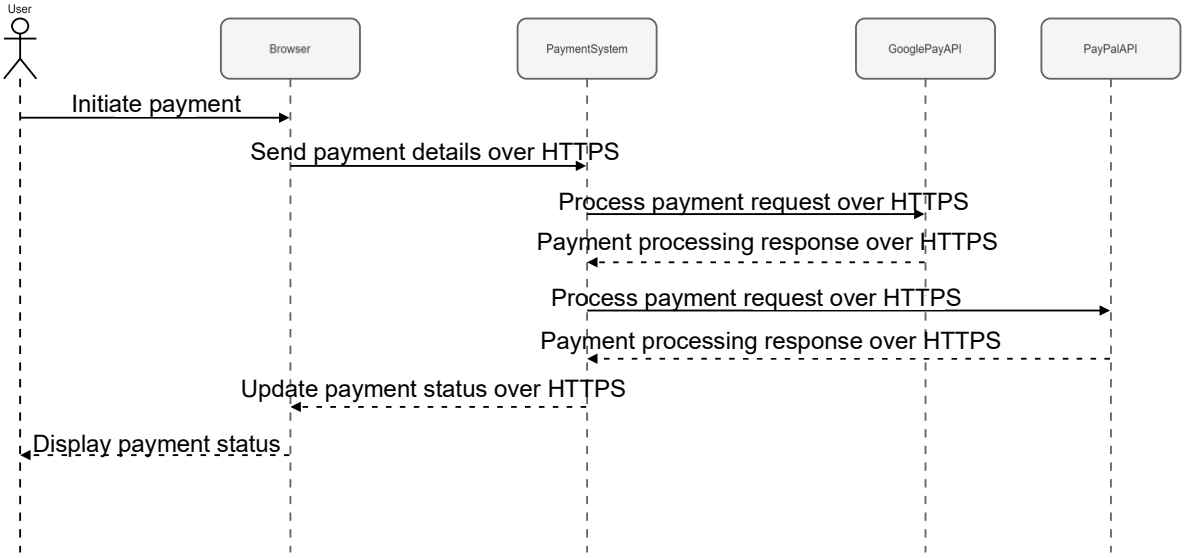
Name	View 1.4.1.3.2: Transaction Table
Description	Records transaction details and manages processing.
Purpose	Contains transaction information and provides methods for processing transactions.
Requirements	Database table for storing transaction details and handling transaction processes.
Elements	**TransactionTable**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.3.3: Payment Table



Name	**View 1.4.1.3.3: Payment Table**
Description	Stores payment details and manages payment processing.
Purpose	Contains payment information and provides methods for processing payments.
Requirements	Database table for storing payment details and handling payment processes.
Elements	**PaymentTable**
Referenced By	
Viewpoint	Class Diagram

View 1.4.1.4: Implement HTTPS



Name	View 1.4.1.4: Implement HTTPS
Description	Update the communication protocol to HTTPS for secure interactions with Google Pay and PayPal APIs.
Purpose	Ensure that all communication with the Google Pay API and PayPal API is secured using HTTPS.
Requirements	A system capable of establishing secure HTTPS connections for API interactions and ensuring data integrity.
Elements	**User,** **Browser,** **PaymentSystem,** **Google Pay API,** **PayPal API**
Referenced By	
Viewpoint	Sequence Diagram

View 1.4.1.6: JSON file request to API

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "RequestID": { "type": "string", "pattern": "^[0-9a-fA-F]{8}$" },
    "Timestamp": {
      "type": "integer",
      "description": "Unix timestamp in milliseconds",
      "minimum": 1718390400000, "maximum": 253402300799999
    },
    "PaymentAmount": { "type": "number", "minimum": 0 },
    "Currency": { "type": "string", "pattern": "^[A-Z]{3}$" },
    "PaymentMethod": { "type": "string", "enum": ["GooglePay", "PayPal"] },
    "MerchantID": { "type": "string", "pattern": "^[0-9a-zA-Z]{8,20}$" },
    "CustomerID": { "type": "string", "pattern": "^[0-9a-zA-Z]{8,20}$" },
    "CallbackURL": { "type": "string", "pattern": "^(http|https)://" },
    "Signature": { "type": "string", "description": "HMAC-SHA256 signature" }
  },
  "required": ["RequestID", "Timestamp", "PaymentAmount", "Currency", "PaymentMethod", "MerchantID",
"CustomerID", "CallbackURL", "Signature"]
}
```

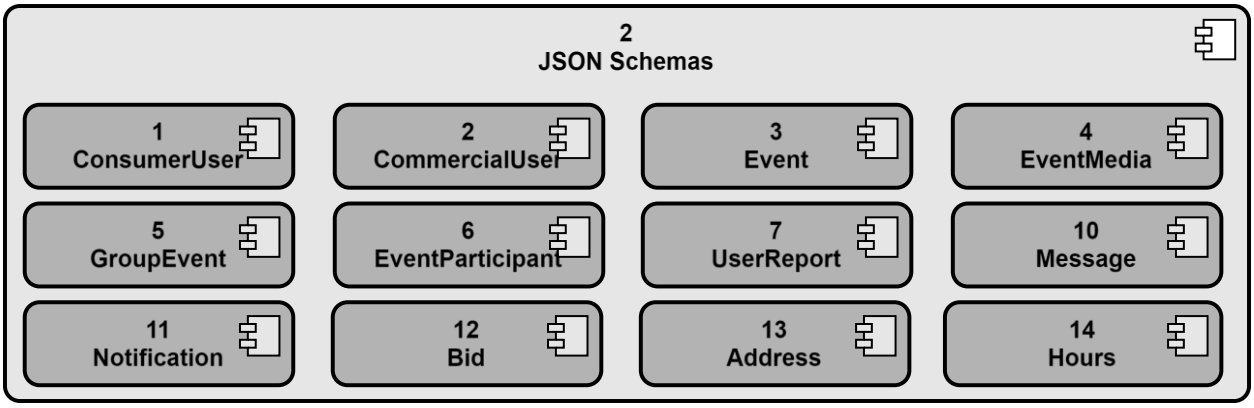
Name	View 1.4.1.6: JSON file request to API
Description	Facilitate communication between the client and the payment APIs using JSON file requests and handling the responses.
Purpose	Send a JSON file request to GooglePay API and PayPal API for payment processing.
Requirements	A system capable of creating, sending, and processing JSON file requests to payment APIs.
Elements	**JSON Request File**
Referenced By	
Viewpoint	JSON Schema

View 1.4.1.7: JSON file response from APIs

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "ResponseID": { "type": "string", "pattern": "^[0-9a-fA-F]{8}$" },
    "Timestamp": {
      "type": "integer",
      "description": "Unix timestamp in milliseconds",
      "minimum": 1718390400000, "maximum": 253402300799999
    },
    "Status": { "type": "string", "enum": ["Success", "Failure", "Pending"] },
    "Message": { "type": "string" },
    "TransactionID": { "type": "string", "pattern": "^[0-9a-zA-Z]{8,20}$" },
    "PaymentAmount": { "type": "number", "minimum": 0 },
    "Currency": { "type": "string", "pattern": "^[A-Z]{3}$" },
    "CustomerID": { "type": "string", "pattern": "^[0-9a-zA-Z]{8,20}$" },
    "MerchantID": { "type": "string", "pattern": "^[0-9a-zA-Z]{8,20}$" },
    "Signature": { "type": "string", "description": "HMAC-SHA256 signature" }
  },
  "required": ["ResponseID", "Timestamp", "Status", "Message", "TransactionID", "PaymentAmount", "Currency",
"CustomerID", "MerchantID", "Signature"]
}
```

Name	View 1.4.1.7: JSON file response from APIs
Description	Handle and structure the responses received from GooglePay API and PayPal API for client processing.
Purpose	Return a JSON file response from GooglePay API and PayPal API for payment processing.
Requirements	System capable of handling, structuring, and validating JSON file responses.
Elements	**JSON Response File**
Referenced By	
Viewpoint	JSON Schema

View 2: JSON Schemas



Name	2 Schemas
Purpose	Enumerate the various JSON schemas the project will use
Description	This is a list of JSON Schemas the project will use. As they are use across the Uvents project, they are placed at a high level. JSON schemas will be used to specify the data and format the client and server will use to communicate. There will also be schemas for 3rd party API interactions.
Requirements	
Elements	2.1 **ConsumerUser** User not affiliated with a company or other organization
	2.2 **CommercialUser** Accounts associated with a company or other organization
	2.3 **Event** Commercial user event
	2.4 **EventMedia** Media resources associated with events
	2.5 **GroupEvent** Consumer group event
	2.6 **EventParticipant** Table linking consumer users to events they are attending
	2.7 **UserReport** For reporting undesirable user activity
	2.10 **Message** For consumer user to send messages to eachother
	2.11 **Notification** A simple, generic notification
	2.12 **Bid** Commercial users bid for presence on the site or app
	2.13 **Address** For storing addresses of events
	2.14 **Hours** For storing event hours
Referenced by	2
Viewpoint	Component Diagram

View 2.1: Consumer User JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "ConsumerUser",
    "type": "object",
    "properties": {
        "FirstName": {
            "type": "string",
            "description": "First name of the consumer user"
        },
        "LastName": {
            "type": "string",
            "description": "Last name of the consumer user"
        },
        "Role": {
            "type": "integer",
            "description": "Role enumeration. [consumerUser,commercialUser,adminUser]. TODO: Is this
redundant?"
        },
        "Username": {
            "type": "string",
            "description": "Username used by the consumer user for login purposes"
        },
        "SocialMediaID": {
            "type": "string",
            "pattern": "^[0-9a-fA-F]{8}$",
            "description": "8-character hexadecimal identifier for social media integration"
        },
        "Password": {
            "type": "string",
            "minLength": 10,
            "pattern": "^(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*(),.?\":{}|<>]).{6,}$",
            "description": "Password must be at least 6 characters long, contain one special character,
one uppercase letter, and one number"
        },
        "PasswordReset": {
            "type": "boolean",
            "description": "Flag indicating if the consumer user's password has been reset"
        },
        "ResetTokenExpiration": {
            "type": "integer",
            "description": "Timestamp in milliseconds indicating when the password expires",
            "minimum": 1718390400000,
            "maximum": 253402300799999
        },
        "Email": {
            "type": "string",
            "pattern": "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
            "description": "Email address of the consumer user"
        },
        "PhoneNumber": {
            "type": "string",
            "pattern": "^\\d{3}-\\d{3}-\\d{4}$",
            "description": "Phone number of the consumer user"
        },
        "EventsAttended": {
            "type": "integer",
            "description": "Total number of events attended by the consumer user"
        },
        "EventsHosted": {
            "type": "integer",
            "description": "Total number of events hosted by the consumer user"
        },
        "Gender": {
            "type": "integer",
            "description": "Gender identifier of the consumer user"
```

```
        }
    },
    "required": [
        "FirstName",
        "LastName",
        "Role",
        "Username",
        "SocialMediaID",
        "Password",
        "PasswordReset",
        "ResetTokenExpiration",
        "Email",
        "PhoneNumber",
        "EventsAttended",
        "EventsHosted",
        "Gender"
    ]
}
```

Name	2.1 Consumer User JSON Schema
Purpose	Define the JSON schema for consumer user data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses.
Requirements	SAS.2.5, FV.1.5, URV.2, URV.2.8
Elements	**None**
Referenced by	2
Viewpoint	JSON Schema

View 2.2: Commercial User JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "CommercialUser",
    "type": "object",
    "properties": {
        "Name": {
            "type": "string",
            "description": "The name of the commercial user"
        },
        "Description": {
            "type": "string",
            "description": "A brief description of the commercial user"
        },
        "Category": {
            "type": "integer",
            "description": "The category ID associated with the commercial user",
            "enum": [1, 2, 3, 4, 5]
        },
        "Address": {
            "type": "string",
            "description": "The address of the commercial user"
        },
        "Phone": {
            "type": "string",
            "description": "The phone number of the commercial user",
            "pattern": "^[0-9]{3}-[0-9]{3}-[0-9]{4}$"
        },
        "Email": {
            "type": "string",
            "description": "The email address of the commercial user",
            "format": "email"
        },
        "Website": {
            "type": "boolean",
            "description": "Is there a website?"
        },
        "Picture": {
            "type": "integer",
            "description": "Timestamp of the picture associated with the commercial user",
        },
        "SocialMedia": {
            "type": "string",
            "description": "The social media profile of the commercial user"
        },
        "ThirdPartyPayment": {
            "type": "string",
            "description": "The third-party payment system used by the commercial user. Paypal/Google
pay"
        },
        "Verified": {
            "type": "integer",
            "description": "Verification status ID of the commercial user"
        }
    },
    "required": [
        "Name",
        "Description",
        "Category",
        "Address",
        "Phone",
        "Email",
        "Website",
        "Picture",
        "SocialMedia",
        "ThirdPartyPayment",
```

```
        "Verified"
    ]
}
```

Name	2.2 Commercial User JSON Schema
Purpose	Define the JSON schema for commercial user data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses.
Requirements	PO.3.2, PO.3.5, FS.15,
Elements	**None**
Referenced by	2
Viewpoint	JSON Schema

View 2.3: Event JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Event",
    "type": "object",
    "properties": {
        "OrganizerId": {
            "type": "integer",
            "description": "The ID of the event organizer"
        },
        "Title": {
            "type": "string",
            "description": "The title of the event"
        },
        "Description": {
            "type": "string",
            "description": "Event description"
        },
        "Price": {
            "type": "number",
            "description": "The price of the event",
            "minimum": 0
        },
        "Icon": {
            "type": "string",
            "description": "The resource locator of the event icon. TODO: How are they stored?"
        },
        "AddressId": {
            "type": "integer",
            "description": "The ID of the event address"
        },
        "StartTime": {
            "type": "integer",
            "description": "Timestamp indicating the start of the event"
        },
        "EndTime": {
            "type": "integer",
            "description": "Timestamp indicating the end of the event"
        },
        "Active": {
            "type": "boolean",
            "description": "Is the event currently active"
        }
    },
    "required": [
        "OrganizerId",
        "Title",
        "Description",
        "Price",
        "Icon",
        "AddressId",
        "StartTime",
        "EndTime",
        "Active"
    ]
}
```

Name	2.3 Event JSON Schema
Purpose	Define the JSON schema for event data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. This represents the data stored for events.

Requirements	PO.3.4.3, PO.3.5.3, EIS.1.4, FS.2, LDR.3
Elements	2.2 **OrganizerId:** Commercial user that created the event.
	2.13 **AddressId:** Id for the address record where the event takes place.
Referenced by	2
Viewpoint	JSON Schema

View 2.4: EventMedia JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "EventMedia",
    "type": "object",
    "properties": {
        "EventId": {
            "type": "integer",
            "description": "The ID of the event this media belongs to."
        },
        "Media": {
            "type": "string",
            "description": "The event media data. TODD: is this a URL or a Base64-encoded image."
        }
    },
    "required": ["EventId", "Media"]
}
```

Name	2.4 EventMedia JSON Schema
Purpose	Define the JSON schema for event media data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. This represents the media data stored for events, such as photos or videos.
Requirements	LDR.4, LDV.4
Elements	2.3 **EventId:** Index to the event the media is for.
Referenced by	2
Viewpoint	JSON Schema

View 2.5: GroupEvent JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "GroupEvent",
    "type": "object",
    "properties": {
        "OrganizerId": {
            "type": "integer",
            "description": "The index of the consumer user who created the event"
        },
        "StartTime": {
            "type": "integer",
            "description": "Timestamp indicating the start time of the event"
        },
        "EndTime": {
            "type": "integer",
            "description": "Timestamp indicating the end time of the event"
        },
        "Description": {
            "type": "string",
            "description": "Description of the group event"
        },
        "AddressId": {
            "type": "integer",
            "description": "The index of the address where the group event occurs"
        },
        "IsDate": {
            "type": "boolean",
            "description": "TODO: Is dating permitted?"
        },
        "MaxParticipants": {
            "type": "integer",
            "description": "Maximum participants allowed at the event"
        }
    },
    "required": [
        "OrganizerId",
        "StartTime",
        "EndTime",
        "Description",
        "AddressId",
        "IsDate",
        "MaxParticipants"
    ]
}
```

Name	2.5 GroupEvent JSON Schema
Purpose	Define the JSON schema for group event data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. This represents the data stored for a group event.
Requirements	LDR.5, FV.3, FV.8, URV.4.6, URV.4.7, URV.4.9, LDV.5
Elements	2.1 **OrganizerId**: Index of consumer user who created the event.
	2.13 **AddressId**: Address index for the location of the event
Referenced by	2
Viewpoint	JSON Schema

View 2.6: EventParticipant JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "EventParticipant",
    "type": "object",
    "properties": {
        "ConsumerUserId": {
            "type": "integer",
            "description": "The index of the consumer user participant"
        },
        "GroupEventId": {
            "type": "integer",
            "description": "The index of the group event the user is attending"
        },
        "ApprovalStatus": {
            "type": "boolean",
            "description": "Indicates whether the participant has been approved by the event organizer"
        }
    },
    "required": ["ConsumerUserId", "GroupEventId", "ApprovalStatus"]
}
```

Name	2.6 EventParticipant JSON Schema
Purpose	Define the JSON schema for event participant data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. This represents the data stored for a participating consumer user.
Requirements	LDR.6, LDV.6
Elements	2.1 **ConsumerUserId**: Index of consumer user who is attending the event.
	2.5 **GroupEventId**: Index of the group event that is being attended.
Referenced by	2
Viewpoint	JSON Schema

View 2.7: User Report JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "UserReport",
    "type": "object",
    "properties": {
      "ReportingUserID": {
        "type": "integer",
        "description": "The ID of the ConsumerUser who is reporting"
      },
      "ReportedUserID": {
        "type": "integer",
        "description": "The ID of the ConsumerUser being reported"
      },
      "ReportType": {
        "type": "integer",
        "description": "The enumerated type of the report. No Show/Spam/Fake Account"
      },
      "EventID": {
        "type": "integer",
        "description": "The ID of the event the report is referencing"
      }
    },
    "required": ["ReportingUserID", "ReportedUserID", "ReportType", "EventID"]
  }
```

Name	2.7 User Report JSON Schema
Purpose	Define the JSON schema for user report data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses.
Requirements	LDR.7, LDV.7
Elements	2.1 **ReportingUserID, ReportedUserID:** These are indexes to ConsumerUser records
	2.3 **EventID:** This is an index to an Event record
Referenced by	2
Viewpoint	JSON Schema

View 2.10: Message JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "type": "object",
    "properties": {
        "Originator": {
            "type": "integer",
            "description": "The index of the originating ConsumerUser record"
        },
        "ConversationId": {
            "type": "integer",
            "description": "The index of the Conversation record"
        },
        "Content": {
            "type": "string",
            "description": "Message content"
        },
        "Date": {
            "type": "string",
            "format": "date",
            "description": "The date in YYYY-MM-DD format"
        },
        "Time": {
            "type": "string",
            "pattern": "^(?:[01]\\d|2[0-3]):[0-5]\\d:[0-5]\\d$",
            "description": "The time in HH:MM:SS format"
        },
        "Status": {
            "type": "boolean",
            "description": "Message has been successfully sent"
        }
    },
    "required": [
        "Originator",
        "ConversationId",
        "Content",
        "Date",
        "Time",
        "Status"
    ]
}
```

Name	2.10 Message JSON Schema
Purpose	Define the JSON schema for user report data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. It validates messages sent between users.
Requirements	LDR.10, LDV.10, PO.1.2.5, PO.1.7.7, PO.1.7.23, PO.1.7.36, FS.10.3
Elements	**2.1 Originator:** This is an index to a ConsumerUser record
	2.8 ConversationId: This is an index to conversation record
Referenced by	2
Viewpoint	JSON Schema

View 2.11: Notification JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Notification",
    "type": "object",
    "properties": {
        "ConsumerReceiver": {
            "type": "integer",
            "description": "Index to a ConsumerUser table"
        },
        "Type": {
            "type": "integer",
            "enum": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
            "description": "Enumerated type of the notification.",
            "examples": [
                { "value": 1, "description": "Commercial" },
                { "value": 2, "description": "Issue" },
                { "value": 3, "description": "Consumer-Message" },
                { "value": 4, "description": "Consumer-Event" },
                { "value": 5, "description": "Consumer-Friend" },
                { "value": 6, "description": "Consumer-Group" },
                { "value": 7, "description": "Consumer-Nearby" },
                { "value": 8, "description": "Admin-Message" },
                { "value": 9, "description": "Admin-Event" },
                { "value": 10, "description": "Admin-Commercial-Created" },
                { "value": 11, "description": "Admin-Flagged" },
                { "value": 12, "description": "Admin-Ticket" }
            ]
        }
    },
    "required": ["ConsumerReceiver", "Type"]
}
```

Name	2.11 Notification JSON Schema
Purpose	Define the JSON schema for notifications data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. It validates notifications sent to users about various events.
Requirements	LDR.11, LDV.11, FV.12, FV.36, URV.1.6, URV.1.7-13, PO.1.2.10, PO.1.7.15, PO.1.7.29, PO.1.7.41, PO.2.1.14, PO.2.3.11, FS.12, FS.36
Elements	**2.1 Consumer Receiver:** This is an index to the ConsumerUser record the notification is being sent to
Referenced by	2
Viewpoint	JSON Schema

View 2.12 Bid JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Bid",
    "type": "object",
    "properties": {
        "BusinessId": {
            "type": "integer",
            "description": "Index to a CommercialUser record"
        },
        "Month": {
            "type": "integer",
            "enum": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
            "description": "Month the bid was created",
            "examples": [
                { "value": 1, "description": "January" },
                { "value": 2, "description": "February" },
                { "value": 3, "description": "March" },
                { "value": 4, "description": "April" },
                { "value": 5, "description": "May" },
                { "value": 6, "description": "June" },
                { "value": 7, "description": "July" },
                { "value": 8, "description": "August" },
                { "value": 9, "description": "September" },
                { "value": 10, "description": "October" },
                { "value": 11, "description": "November" },
                { "value": 12, "description": "December" }
            ]
        },
        "Year": {
            "type": "integer",
            "minimum": 1900,
            "description": "Year the bid was created"
        },
        "Amount": {
            "type": "number",
            "minimum": 0,
            "description": "Bid amount"
        },
        "Paid": {
            "type": "boolean",
            "description": "Payment status of the bid"
        }
    },
    "required": ["BusinessId", "Month", "Year", "Amount", "Paid"]
}
```

Name	2.12 Bid JSON Schema
Purpose	Define the JSON schema for bid data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. It validates bids created by a commercial user to determine advertising preference
Requirements	LDR.12, EIV.1.6, FV.23, LDV.12, PO.3.2, EIS.1.6, FS.23.4
Elements	2.2 **BusinessId:** This is an index to the CommercialUser record that bid is associated with.
Referenced by	2
Viewpoint	JSON Schema

View 2.13: Address JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Address",
    "type": "object",
    "properties": {
        "Originator": {
            "type": "integer",
            "description": "Index referencing a ConsumerUser record"
        },
        "Line1": {
            "type": "string",
            "description": "Street address line 1"
        },
        "Line2": {
            "type": "string",
            "description": "Street address line 2 (optional)"
        },
        "City": {
            "type": "string",
            "description": "City of the address"
        },
        "StateId": {
            "type": "integer",
            "description": "Index referencing State record. TODO. find table or algorithm for this
reference"
        }
    },
    "required": ["Originator", "Line1", "City", "StateId"]
}
```

Name	2.13 Address JSON Schema
Purpose	Define the JSON schema for address data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. It validates addresses
Requirements	LDR.3.7, LDR.13, LDV.13
Elements	2.1 **Originator:** This is an index to the ConsumerUser record the address is associated with
	X.X StateId: Index, enumeration or algorithm for states.
Referenced by	2
Viewpoint	JSON Schema

View 2.14: Hours JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Hours",
    "type": "object",
    "properties": {
        "BusinessId": {
            "type": "integer",
            "description": "Index referencing a CommercialUser record"
        },
        "Open": {
            "type": "boolean",
            "description": "List whether a slot is open to be booked for an event"
        },
        "Day": {
            "type": "integer",
            "description": "List open days that events can take place on."
        },
        "Time": {
            "type": "integer",
            "description": "List the time of the event. This is a timestamp"
        }
    },
    "required": ["BusinessId", "Open", "Day", "Time"]
}
```

Name	2.14 Hours JSON Schema
Purpose	Define the JSON schema for hours data
Description	This schema is useful for validating data sent as JSON objects sent in the bodies of http requests and responses. It validates the hours of operation for its associated event
Requirements	LDR.14, LDV.14
Elements	2.2 **BusinessId:** This is an index to the CommercialUser record the hours are associated with
	X.X Day: Index, enumeration or algorithm for day. (currently undefined)
Referenced by	2

Requirements Traceability Matrix

Requirement Location	Requirement Description	View #
FS.1	Create New Account will allow the consumer to create their personalized profile.	1.2.2.1 1.2.2.1.1
FS.4	The Edit Profile page will allow the consumer to make changes to their profile.	1.2.2.1 1.2.2.1.4
FS.5	View My profile will allow the Consumer User to view the information within their profile.	1.2.2.1 1.2.2.1.3
FS.6	The Login will allow authorized access to the consumer.	1.2.2.1 1.2.2.1.2 1.2.2.1.2.1 1.2.2.1.2.2
FS.7	The Discover page will recommend events to consumers based on their preferences.	1.2.2.1 1.2.2.1.6 1.2.2.1.13
FS.8	The Create Group Event feature will allow the consumer to create events based on their specifications.	1.2.2.1 1.2.2.1.9
FS.9	Edit Event will allow the consumer to customize the events created by them.	1.2.2.1
FS.10	This feature shall allow the consumer to communicate.	1.2.2.1 1.2.2.1.14
FS.11	My Events shall group the list of events that the consumer is linked to.	1.2.2.1 1.2.2.1.8
FS.13	Settings will allow the consumer to modify the configuration of the app to their liking.	1.2.2.1 1.2.2.1.5
FS.15	Create New Account will allow business owners to create an account.	1.2.2.1.1
FS.15.3	The create new account feature shall validate the information entered into by the business users.	1.2.1.3.2
FS.17	View My profile will allow the Commercial User to view the information within their profile.	1.2.2.1.3
FS.18	Login will allow authorized access to the business owners.	1.2.2.1.2 1.2.2.1.2.1 1.2.2.1.2.2
FS.19	Create Event will allow the commercial user to create events based on their specifications.	1.2.2.1.9
FS.20	Manage Commercial Event will display options for managing an event.	1.2.2.1.10
FS.21	Past Events will allow the user to view their past events.	1.2.2.1
FS.23	Promote event will allow a commercial user to view advertising space auctions.	1.2.2.1 1.2.2.1.15.1
FS.24	Settings will allow the commercial user to modify the configuration of the app to their liking.	1.2.2.1.5
FS.31.2	The Create New Admin function shall validate the information entered into by the admin users.	1.2.1.3.2
FS.36	Notifications will allow the admin to receive notifications/alerts.	1.2.2.1.14.1
PO 1.2.1	Uvents will provide a Login interface that will send the contents from a form to the server. The required elements of the form will be passed to the Login operation on the server.	1.2.1.3
PO 1.2.2	Uvents will provide New Account Creation option.	1.2.1.3
PO 1.2.3	Uvents will provide a Logout option.	1.2.1.5
PO 1.2.4	Uvents will provide an Event Creation option (consumer, commercial).	1.2.1.7.1
PO 1.2.5	Uvents will provide a Message option.	1.2.1.6
PO 1.2.6	Uvents will provide an Edit Profile option.	1.2.1.4
PO 1.2.7	Uvents will provide an Edit Event option.	1.2.1.7.4
PO 1.2.8	Uvents will provide a Search Bar option.	1.2.1.7.2.2
PO 1.2.9	Uvents will provide a Filter Event option.	1.2.1.7.2.2
PO 1.2.11	Uvents will provide a Settings option.	1.2.1.5
PO 1.2.12	Uvents will have a Home View with easy access to key information	1.2.1.2

PO 1.2.13	Uvents will have a View for Past Events.	1.2.1.7.2
PO 1.2.14	Uvents will provide an Event Feed. (consumer, admin]	1.2.1.7.2.2
PO 1.2.15	Uvents will provide interface for user to view profile.	1.2.1.4.1
PO 1.2.18	Uvents will provide an option for Admins to edit other users.	1.2.4.7
PO 1.7.3	The application will allow users to view and edit profile information.	1.2.1.4.2
PO 1.7.5	The application will allow for discovery of events (with or without promotions) through a feed.	1.2.1.7.2.2
PO 1.7.6	The application will allow a user to view events they created.	1.2.1.7.2
PO 1.7.7	The application will allow consumers to send direct messages to each other.	1.2.1.6
PO 1.7.8	The application will allow users to create events and post them.	1.2.1.7.1
PO 1.7.9	The application will allow users to edit events which they have created.	1.2.1.7.4
PO 1.7.10	The application will allow users to delete events which they have created.	1.2.1.7.4
PO 1.7.11	The application will allow users to copy a previous event and use it for a current event.	1.2.1.7.4
PO 1.7.16	The application will allow consumers to edit various settings of the application.	1.2.1.5
PO 1.7.17	Consumer users will be provided with a home screen view. that will provide quick access to events and other options.	1.2.1.2
PO 1.7.21	Commercial users will be provided with a home screen view.	1.2.1.2
PO 1.7.22	Commercial users will be able to view all previous events they had created.	1.2.1.7.2
PO 1.7.23	Commercial users will have the ability to communicate with consumer users via the messaging feature. The message feature will prove instrumental for commercial users to answer any questions the consumer may have and provide a channel for them to receive feedback about their events.	1.2.1.6.1
PO 1.7.24	Commercial users will be able to edit their account information including their username and profile picture.	1.2.1.4.2
PO 1.7.25	Commercial users will be able to create events to be listed for consumers to use in their group events.	1.2.1.7.1
PO 1.7.26	Commercial users will be able to edit the events they have created.	1.2.1.7.4
PO 1.7.31	Admin shall login from the same login view as the consumer.	1.2.1.3
PO 1.7.32	The creation of an Admin account will be accomplished only with the apporval of an already authorized admin account.	1.2.4.1
PO 1.7.33	Admin will be able to logut of their account.	1.2.2.1
PO 1.7.34	Admin will be provided with a home screen view will provide quick and simple access to Admin tools.	1.2.1.2
PO 1.7.35	Admin will be able to read, archive, copy and edit past events.	1.2.1.7.4, 1.2.4.5
PO 1.7.36	Admin will be able to message Consumer and Commercial users freely.	1.2.1.6, 1.2.4.2
PO 1.7.37	Admin will be able to edit their profile.	1.2.4.9 1.2.1.4.1.3
PO 1.7.38	Admin will able to access and edit both Consumer and Commercial accounts.	1.2.4.7
PO 1.7.39	Admin will have a tool to filter and sort users or events.	1.2.4.10, 1.2.1.7.2.2
PO 1.7.40	Admin will have access to a search bar to search for specific users or events.	1.2.1.7.2.2
PO 1.7.41	Admin will receive notifications of issues in need of address.	1.2.4.4
PO 1.7.42	Admin will have access to settings to personalize thier account.	1.2.4.6 1.2.1.4.1.3
PO 2.1.4	Edit Profile – FS.4 The Edit Profile page will allow the consumer to make changes to their profile.	1.2.1.4.1
PO 2.1.6	Consumer Login	1.2.2.1.2.1 1.2.2.1.2.2
PO 2.1.12	Event Messaging – FS.10 This feature shall allow the consumer to communicate.	1.2.1.6.1.1
PO 2.1.13	My Events – FS.11 My Events shall group the list of events that the consumer is linked to.	1.2.1.7.2
PO 2.2.6	Business Information – FS.16.1 Business Information will allow a business' general information to be altered.	1.2.1.4.1

PO 2.2.7	Payment Information – FS.16.2 Payment Information will allow a business' payment information to be altered.	1.2.1.4.1
PO 2.2.9	Commerical Login	1.2.2.1.2.1 1.2.2.1.2.2
PO 2.2.14	Past Events – FS.21 Past Events will allow the user to view their past events	1.2.1.7.2
PO 2.2.17	Start Auction	1.2.2.1.15.1
PO 2.3.1	View Commercial Reports - FS. 27 Commercial Reports will allow the admin to view reports of consumer user content.	1.2.4.3
PO 2.3.2	View consumer Reports - FS. 26 Consumer Reports will allow the admin to view reports of consumer user content.	1.2.4.3
PO 2.3.3	Edit Profile - FS. 28 Profile Page will contain user information about the admin	1.2.4.9 1.2.1.4.1
PO 2.3.4	View my profile - FS 29 View My profile will allow the Admin User to view the information within their profile.	1.2.4.9.1, 1.2.1.4.1.3
PO 2.3.5	Admin Login - FS. 30 The Login function will allow authorized access to the admin.	1.2.1.3
PO 2.3.6	Create new Admin - FS. 31 Create New Admin function will allow authorized Admin accounts to be created.	1.2.4.1
PO 2.3.7	Edit Users - FS. 32 Edit Users will give the admin the ability to edit consumer profiles.	1.2.4.7 1.2.1.4.1
PO 2.3.8	Edit Events - FS. 33 Edit Events will give the admin all abilities regarding event editing.	1.2.4.5, 1.2.4.8
PO 2.3.9	Message - FS 34. Message allow the admin to send/receive messages.	1.2.4.2 1.2.1.6.1.3
PO ~~2.3.10~~	Design - FS.35 Design will allow the admin users to easily navigate through the app.	
PO 2.3.11	Notifications - FS. 36 Notifications will allow the admin to receive notifications/alerts.	1.2.4.4 1.2.2.14.1
PO 2.3.12	Settings - FS. 37 Settings will allow the admin to modify the configuration of the app to their liking.	1.2.4.6
PO 2.3.13	Logout - FS. 38 Logout will allow the admin to securely sign off from the app.	1.2.2.1.5.2
PO 2.3.14	View Consumer Profile - FS. 39 View Consumer Profile will allow information with the consumer profile to be viewed.	1.2.4.10, 1.2.1.4.1.2
PO 2.3.15	View Commercial Profile - FS. 40 View Consumer Profile will allow information with the consumer profile to be viewed.	1.2.4.10, 1.2.1.4.1.1
PO 3.4.2	Home Screen Workflow – The process the consumer user will follow when editing profile information and seeing joined group events or messages.	1.2.1.2.1
PO 3.5	The Uvent Database shall have all tables necessary to store information required by the application.	1.1.1
PO 3.5.1	The user table shall store all information about a given user and shall be referred to by other tables that involve users.	1.1.1.1, 1.1.1.1.1
PO 3.5.1.1	Consumer User table shall contain the user ID column as the primary key.	1.1.1.1, 1.1.1.1.1
PO 3.5.1.2	Consumer User table shall contain a first name column	1.1.1.1, 1.1.1.1.1
PO 3.5.1.3	Consumer User table shall contain a last name column	1.1.1.1, 1.1.1.1.1
PO 3.5.1.4	Consumer User table shall contain a role column	1.1.1.1, 1.1.1.1.1
PO 3.5.1.5	Consumer User table shall contain a username	1.1.1.1, 1.1.1.1.1
PO 3.5.1.6	Consumer User table shall contain an optional column for a Consumer User's social media user ID	1.1.1.1, 1.1.1.1.1
PO 3.5.1.7	Consumer User table shall contain the user's hashed password	1.1.1.1, 1.1.1.1.1
PO 3.5.1.8	Consumer User table shall contain a password reset column	1.1.1.1, 1.1.1.1.1
PO 3.5.1.9	Consumer User table shall contain an expiration date for the reset password token column	1.1.1.1, 1.1.1.1.1
PO 3.5.1.10	Consumer User table shall contain the Consumer User's Email address	1.1.1.1, 1.1.1.1.1
PO 3.5.1.11	Consumer User table shall contain the user's phone number	1.1.1.1, 1.1.1.1.1
PO 3.5.1.12	Consumer User table shall contain a column indicating the number of events the user has attended	1.1.1.1, 1.1.1.1.1
PO 3.5.1.13	Consumer User table shall contain a column indicating the number of events the user has hosted	1.1.1.1, 1.1.1.1.1
PO 3.5.1.14	Consumer User table shall contain a gender column	1.1.1.1, 1.1.1.1.1

PO 3.5.2	Commercial User Table will store all the information needed to represent a given Commercial User	1.1.1.2 1.2.1.2.1.2 1.1.1.2.1
PO 3.5.2.1	Commercial User table shall have a business ID column as the primary key	1.1.1.2, 1.1.1.2.1
PO 3.5.2.2	Commercial User table shall have a name column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.3	Commercial User table shall have a description column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.4	Commercial User table shall have a category column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.5	Commercial User table shall have an address column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.6	Commercial User table shall have a phone column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.7	Commercial User table shall have an email column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.8	Commercial User table shall have a website column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.9	Commercial User table shall have a picture column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.10	Commercial User table shall have a Social Media column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.11	Commercial User table shall have a third party payment column	1.1.1.2, 1.1.1.2.1
PO 3.5.2.12	Commercial User table shall have a verified column	1.1.1.2, 1.1.1.2.1
PO 3.6.2	Home Screen Workflow – The process followed by admin user when editing or deleting events and group events created by commercial and consumer users and when editing or deleting consumer user or commercial user accounts.	1.2.1.2.1
LDR.3	The event table shall store all information about Events and shall be referred to by other tables involving Events.	1.2.2.1.2.7.1
LDR.11	The notification table will include all pertinent information about notification ID column as the primary key.	1.2.2.1.14.1
LDV.1.11	Verify that the Consumer User table shall contain a password reset	1.2.1.3.3
FV.23	Promote Event shall display the time till next bidding auction and shall display button to start auction only when auction is currently taking place.	1.2.2.1.15.1
FV.23.4	Start Auction shall display tabletop ranks of bidding from the users. Shall display two buttons which are + and – to increase or decrease the bid by $5. Shall display place bid button for users to place their bid and see their rank on the table.	1.2.2.1.15.1
F S . 1 5 . 5 . 1	A third-party service will be used for payments	1.4.1
F V . 1 6 . 2 . 1	Payment information will be stored by 3rd party not Uvents	1.4.1.3.1
EIS.5 PayPal	Paypal will be used as a third-party payment service	1.4.1.2
EIS.6 Google Pay	Google will be used as a third-party payment service	1.4.1.1
EIS.2	Consumer users shall have the ability to share events on the Facebook platform if logged into the user's Facebook account	1.3 1.3.1 1.3.1.1 1.3.1.4 1.3.1.5 1.3.1.6 1.3.2 1.3.2.1 1.3.2.2
EIS.3	Consumer users shall have the ability to share events on the Instagram platform if logged into the user's Instagram account	1.3 1.3.1 1.3.1.2 1.3.1.4 1.3.1.5 1.3.1.6 1.3.2 1.3.2.1 1.3.2.2
EIS.4	Consumer users shall have the ability to share events on the Twitter(X) platform if logged into the user's Twitter(X) account	1.3 1.3.1 1.3.1.4 1.3.1.5 1.3.1.6 1.3.2 1.3.2.1 1.3.2.2

URS 1.6	Consumer Users shall receive message notifications.	1.2.1.2.3
URS 1.11	Admin Users shall receive message notifications.	1.2.1.2.3