Latvian University

Faculty of Computing

GigStory

Practical work report for course DatZ2072 “Software Engineering”

Authors:

Dita Ābola (da21002),

Karyna Yakimovich (ky21001),

Eduards Leonovics (el16022),

Madara Lukasevica (ml07010),

Lilite Sadovska (ls09364),

Dana Tumane (dt21003)

Advisor: Mg. sc. comp. Jānis Iļjins

Rīga, 2021

Abstract

Project work describes mobile and web application “GigStory” which will be an insider tool for artists and their managers to keep a log of artists’ gig history and give them and other music industry players (agents, promoters, venues) insights for future tour planning.

Currently musicians and managers are using different applications existing in the market for concert planning and scheduling, however the strong point of this application is it’s focus on gathering and storing key data points about the occurred events for analysis, reporting and future prospecting.

The document will have a general description of the application, database models and design, description of main modules (User Management, Artist Management, Gig management, Venue management, Merchandise management, Announcement management, Report management), functions and requirements, non-functional requirements and user interface concepts. The document will be developed according to the ISO/IEC/IEEE 29148:2018 standard.

Keywords: gig logging system, gig data, artist, manager, venue, tour.

Anotācija

Projekta darbs apraksta "GigStory" mobilo un tīmekļa lietojumprogrammu, kas paredzēta mūziķu un menedžeru vajadzībām, lai reģistrētu notikušos koncertus un izmantotu uzkrātos datus turpmāku koncertturneju plānošanai.

Mūziķi un menedžeri jau izmanto dažādas tirgū esošās aplikācijas koncertu un grafiku plānošanai, bet šīs aplikācijas stiprā puse ir tās fokuss uz būtiskāko datu apkošanu un uzglabāšanu par notikšiem koncertiem, lai tos izmantotu analīzei, pārskatiem un turpmākai koncerturneju organizēšanai.

Dokumentā būs vispārējais programmas apraksts, datu bāzes modeļi un dizains, galveno moduļu (Mākslinieku pārvaldība, Koncertu pārvaldība, Norišu vietu pārvaldība, Suvenīru pārvaldība, Paziņojumu pārvaldība, Pārskatu pārvaldība) apraksts, funkcijas un prasības, nefunkcionālās prasības un lietotāja saskarnes koncepcijas. Dokuments tiks izstrādāts saskaņā ar ISO/IEC/IEEE 29148:2018 standartu.

Atslēgas vārdi: koncertu reģistrēšanas sistēma, koncertu dati, mūziķis, menedžeris, norises vieta, turneja.

Table of Contents

List of figures

Glossary

DBMS	Database management system – a software package designed to define, manipulate, retrieve and manage data in a database
IEC	International Electrotechnical Commission – an international standards organization that prepares and publishes international standards for all electrical, electronic and related technologies
IEEE	Institute of Electrical and Electronics Engineers – a professional association for electronic engineering and electrical engineering
ISO	International Organization for Standardization – an international standard setting body composed of representatives from various national standards organizations
MVC	Model–view–controller – a software design pattern
SQL	Structured Query Language – a domain-specific language used in programming and designed for managing data held in a relational database management system

Introduction

Purpose

The document describes “GigStory” mobile application including a general vision of the application, its users and clients, software requirements specification, restrictions as well as software design specification with partial user interface mockup.

Scope

The “GigStory” mobile application provides comfortable access to gig logging systems to artists and managers making it easy to provide all key information about the ongoing or recently happened gig on the fly in one application. Artists can get a historic gig overview in specific cities/regions/countries, see notes about undesirable concert venues, information about the audience, guests, media representatives who have been in a particular concert, sold tickets and merchandise and other valuable insights.

The application can be used by one artist or by a manager providing services to different artists. The application also can be used by agents, promoters or venue representatives to get insights about a particular artist while exploring reports generated for the particular artist.

Links with other documents

The document will be based on the current version of the standard “Systems and software engineering – Life cycle processes – Requirements engineering” ISO/IEC/IEEE 29148:2018 [2].

Overview

The document will consist of 3 chapters, list of figures and definitions of acronyms and abbreviations.

1. Overall description

Our application will provide end-users with a convenient way to enter, store and compile various data about all artists' concert and tour data in a single location. This will help them in making informed decisions regarding the upcoming tour dates, locations as well as the necessary amount of merchandise and other aspects.

By systematically using the application, they will also gain more leverage for negotiating deals with promoters and music agents and venues profitable for all parties involved and therefore – spend less time and energy planning the tour and gain bigger profits.

1.1. As is

Touring is still one of the most lucrative sections in the music industry. The planning of concert tours can be a challenging process that requires a large amount of data to be analyzed, not only about the audience of the artist in each country/city musician is planning to tour, but also general audience habits, venue characteristics, and also merch sold at concerts, which now makes a big portion of musicians overall income.

For a long time, professionals working in touring had to have a really strong gut feeling when it came to making decisions. In the 21st century musicians worldwide use mobile technologies to help them in all aspects of their everyday professional needs. Data analytics has been one of the biggest changes in music touring. Collecting and evaluating data can give incremental benefits.

In regards to tour management, there are several applications in the market that provide artists and managers with tools to plan upcoming concerts, tours, to-do lists, and other information necessary for concert management, for example, GigPlanner (online calendar, CRM & task manager), which focus on the calendar syncing, task management and other "future events" planning aspects, RoadTrip which focuses on the financial planning and logistics aspects of the tour (schedule, fuel consumption, etc.), Artist Growth which aids the management of the tour information, events, schedule and finances, Merch Cat which monitors sales reports and real-time inventory of the merchandise. However, none of the mentioned applications provide an easy way to log key information about an ongoing or past event, store the data and create reports. But data can serve you as a solid base for planning a tour.

1.2. Client

The system is developed by a group of students as an assignment for the course DatZ2072: Software Engineering. The real client base will be musicians, their managers, and other music industry players, that need to follow the development of their popularity in different regions and the app could be used to get direct insights into the gig history. The app is used by all team members of the crew that are responsible for the show. The generality of the application regarding the data points to register and store will allow this app to be used also by other kinds of traveling entertainers, for example, comedians, traveling circuses, etc.

1.3 Product Vision

The modern world is so used to mobility and quick searches on social networks and applications. That the same representatives of the concert venues answer phone calls with a very small probability. With the advent of our application, you do not need to accumulate personal experience of bad and good venues for concerts, sound, and light teams, now you can find all the reviews from many clients in one place.

With the appearance of the application on the market, artists and managers are provided with convenient access to concert registration systems, which allows them to quickly provide all the key information about the current or recently held concert.

The purpose of this application is to make registration and search for concerts more comfortable and convenient for the seller. To unite many managers in one place, for the exchange of experience, faster search for the necessary sites, and collection of information about the provided service.

The application can be used both by one performer and by a manager providing services to different artists. The app can also be used by agents, promoters or venue representatives to gain insight into a particular artist.

1.4. Business requirements

- A concert can be a standalone concert or part of a tour, public, private or corporate, with or without attendees, guests, and media representatives, with fee for the artist, with or without tickets, with or without sold merchandise.
- It should be possible to enter general non-mandatory information about the gig – name of the gig, whether the gig is part of a tour, whether it is a festival or online gig, sold merchandise, data about the number of attendees, demographics of the attendees, as well

as the type and number of the media (journalists, tv) and/or other invited guests present, as well as notes in free form.

- It should be possible to restrict mandatory registration of the essential data about the concert such as date, time of the day, day of the week, place of the concert such as location (country, city), venue name, venue type, venue capacity.
- It should be possible for the restricted group of users to enter financial data regarding the concert - fee the artist has received, ticket income, income from the sold merchandise, as well as the production costs such as sound, lights, etc. can be registered.
- Location is the city, state, and country where the concert took place. It is not a specific address.
- Gigs that take place in non-public places should be added as happening in private venues. Private venues don't have an address.
- One artist can have several concerts at the same venue on the same day, but starting time should be different.

1.5. System Users

The system is not intended for public band profiles and fans but as an insider tool for artists/managers and other industry players who could use it to gain direct insights into an artist's gig history.

Unregistered users see only the registration page.

There are different roles for registered users – guests, regular users, and admin accounts.

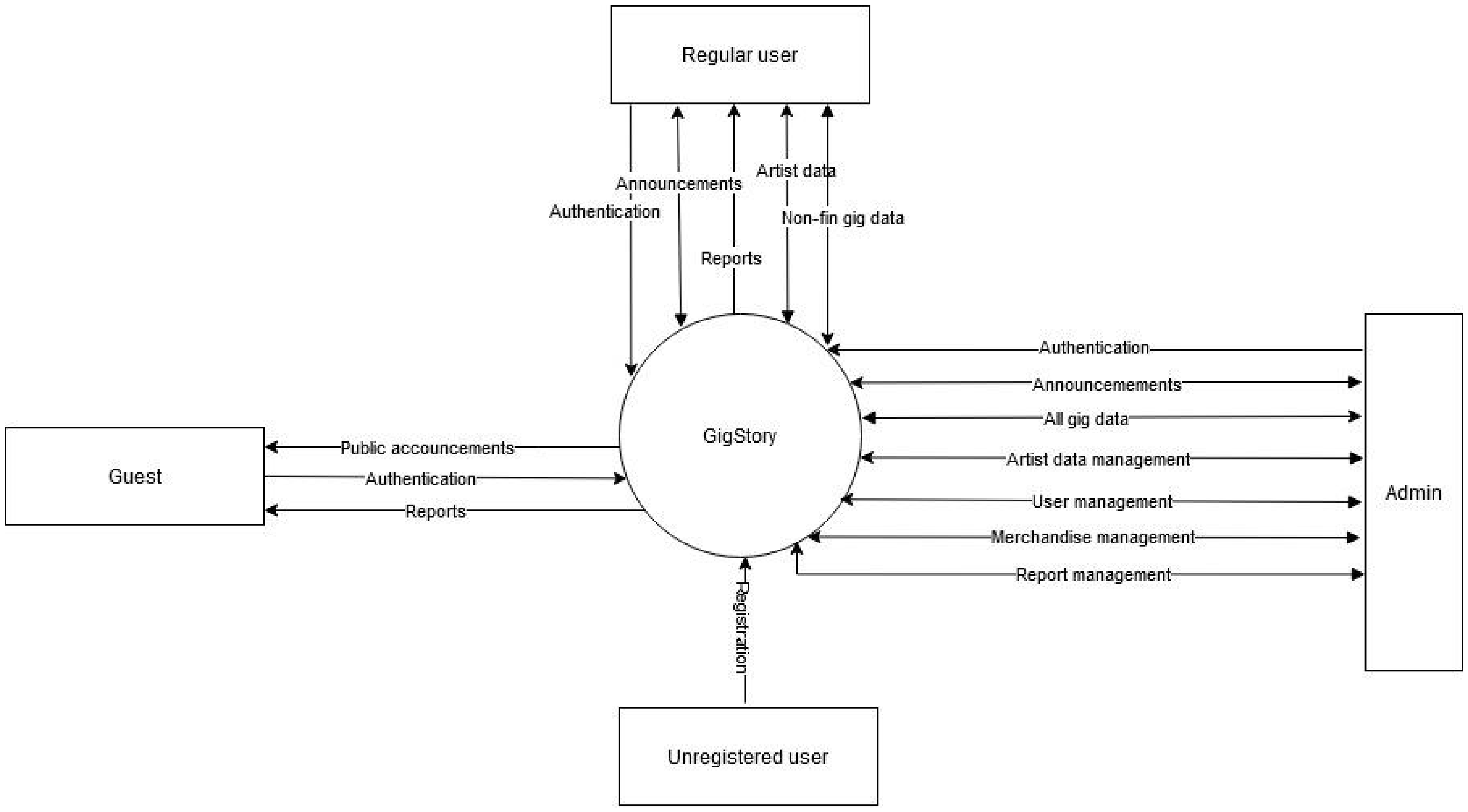
Guest: promoter, agent, venue representative a.o. Registered user, but has only read-only access to the generated gig reports, made available for the particular user, and public announcements.

Regular user: band member, tour manager a.o. Has write access to register and edit gigs (except financial data, for which has only read access). Doesn’t have the right to delete previously saved gigs. Has read access to generated gig reports. Can add items to an already existing merchandise list. Doesn’t have the right to add other users.

Admin: representative of an artist, this role gets assigned to the user who adds an artist account and provides artist email for account verification. Has write access for registering, editing, deleting gigs (also financial data). Admin user can add new users with regular user role. Can generate concert / tour reports. Can create new templates for gig registration to be used by other regular users of the artist. Admin user can create a merchandise list for the artist.

Both regular users and admins have the ability to enter an unlimited number of gigs for one or many artists.

Both regular users and admins can use pre-made templates for registering gigs or registering gigs without a template. For each event, there are mandatory fields (location, date, time, venue type, type of fee received) and optional information fields which can be null (number of attendants, production costs, merch items sold, concert name, tour name, etc.). They can add events in planned/pending/confirmed mode or cancel them.



Pic. 1. Data flow diagram (level 0)

1.6. General restrictions/limitations

As the system is a web and mobile application it’s mandatory for end-users to have a device with a connection to the internet and the application installed. The management company will use cloud-based resources, services, and databases.

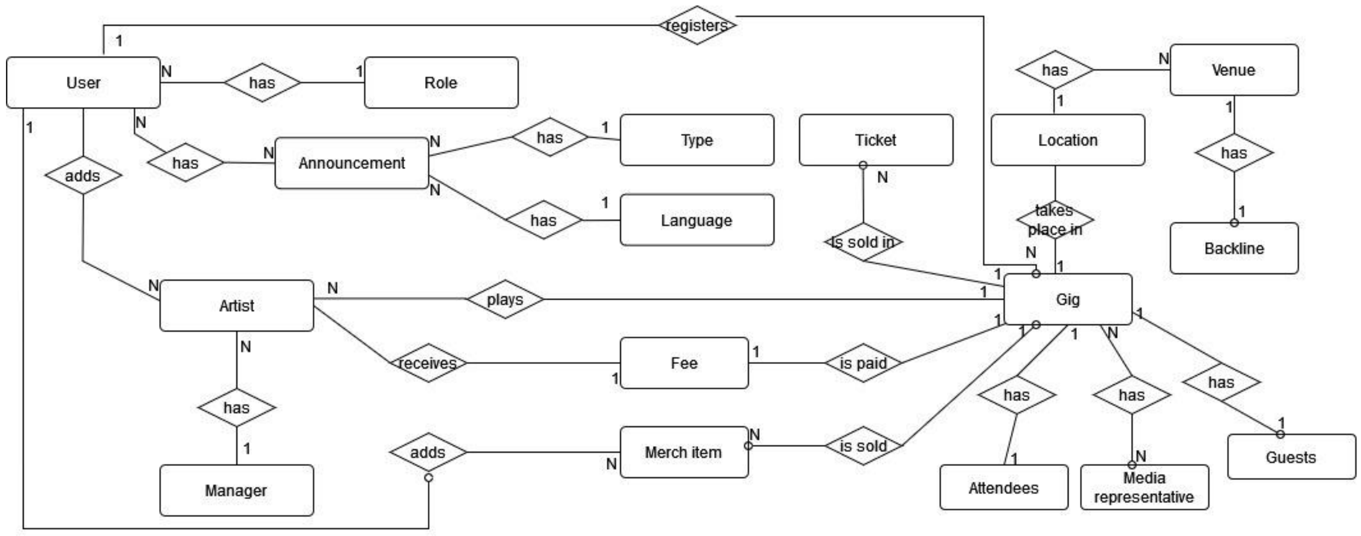
1.7. Assumptions and dependencies

It is assumed that during the installation the users will be added to the database users with the basic permissions to read, write and delete from the database, but only the records created by them, other users’ records can only be read if granted permission. The main dependency of this system will be regarding the authentification, since it will be done by the federation, using either OAuth provider (Google, Facebook).

2. Software requirements specification

2.1. Conceptual Database Model

The conceptual model of the database shows database entities and their connections without attributes and supplementary conjunction table (pic. 2).



Pic. 2. Conceptual database model

Entity “User” has information about user name, password, email, user role, and email notification enabled status. Entity “User” has one-to-many relation with “Artist” entity, meaning that each user with the role “regular user” or “admin” can be related to zero or more artists, and each artist has been created by only one user. Adding an artist entity is not obligatory for a user. Entity “User” has one-to-many relation with “Announcement” entity, a user can have many announcements enabled. Entity “User” has one-to-many relation with “Merch item” entity, meaning that the user can create zero or more merch items, but each merch item is created by one user.

Entity “Role” contains the list of user roles and their descriptions - admin, regular user, guest.

Entity “Announcement” contains information about the messages, users, who created them, and language marks. Entity has relationship with “User” entity.

Entity “Type” is a table containing a list of possible announcement types.

Entity “Language” is a table containing a list of possible languages that can be used for announcements.

Entity “Artist” is a table containing information about the artist name, type, and artist’s manager - entity “Manager”. Entity “Artist” has many-to-one relation with “User” entity, as each regular and admin user can create one or many artists, and each artist can be created only by one user.

Entity “Manager” is a table containing information about a physical person which is an artist's manager. It contains information about a person's name, e-mail, phone number. Each artist can have one manager, but each manager can be manager of one or many artists, there can not be a manager entity that does not have any artists.

Entity “Fee” is a table containing information about the fee that each artist receives about the specific gig. Information is entered when the Gig is getting registered. Entity “Fee” contains information about fee type and amount. Entity has a one-to-one relation with entity “Artist” as one Fee in the table is received by only one artist and only for one gig.

Entity “Merch item” is a table containing information about merch items, their name, type, price and quantity. Related to entity “User” in its creation and to entity “Gig” when it is sold.

Entity “Gig” is a table containing information about each registered gig, its name, status, day of the week, date, year, starting time, artist, fee paid to the artist, production costs, sold tickets, location, venue, is the gig festival, is the gig online, is the recordTemplate, how many attendants, guests and media representatives has the gig had, merch items sold. Entity “Gig” has relationships with entities “User”, “Artist”, “Ticket”, “Fee”, “Merch item”, “Location”, “Attendees”, “Guests”, “Media representative”. It has many-to-one relationships with entity “User”, as each user can register many gigs, but each gig is registered only by one user. It has many-to-one relationship with entity “Artist”, as each registered gig can contain one artist but each artist can have many gigs registered. It has many-to-one relationship with entity “Tickets”, as many tickets can be sold for one gig but each ticket is sold only for one gig.

Entity “Ticket” contains information about the price and quantity of the tickets sold for the gig. It has one-to-many relationship with entity “Gig”.

Entity “Attendees” is a table containing information about the gig’s attendees - their number, average age range, and description. Has one-to-one relationship with entity “Gig”, as each gig can contain one entry of Attendees, and one Attendees entry refers to one Gig.

Entity “Guests” is a table containing information about guests of the gig and contains information about the number of the guests, as well as can contain notes. Has one-to-one relationship with entity “Gig”, as each gig can contain one entry of Guests, and one Guests entry refers to one Gig.

Entity “Media representative” is a table containing information about media representatives - the name of the person, media type, media name. It has many-to-many relationship with entity “Gig”, as each gig can contain many media representatives and each media representative can be registered for many gigs.

Entity “Location” contains information about the city, region, and country. It has many-to-one relationship with entity “Venue”, it can contain zero to many venues.

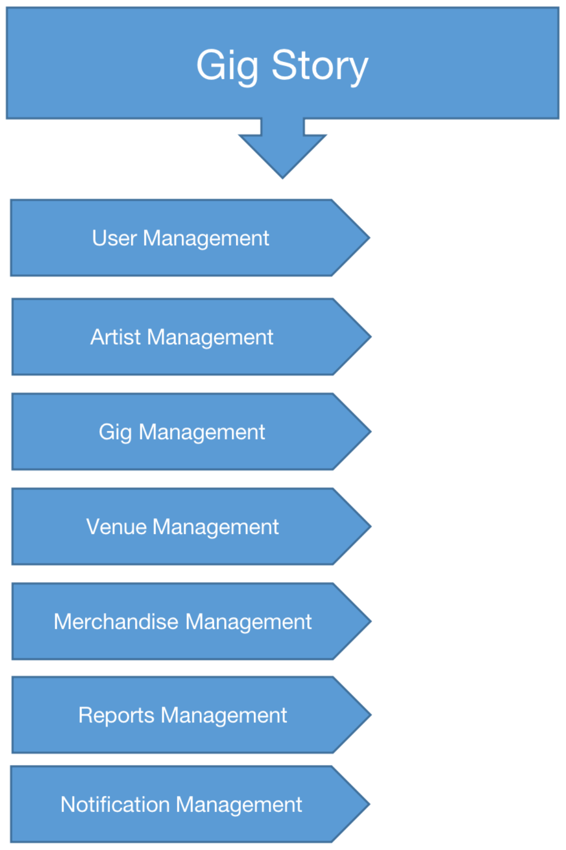
Entity “Venue” contains information about the type, name, the capacity of the venue, as well as can have notes added. It has one-to-many relationship with entity “Location”, as each venue can be in only one Location, but each location can have many venues.

Entity “Backline” contains information about the description and costs of backline equipment in the specific venue. It has zero to one relationship with entity “Venue”, as Venue can have zero or one entry for backline, and each backline is related to only one venue.

2.2. Functional Requirements

2.2.1 General description of functions

The system consists of 7 modules: User Management, Artist Management, Gig management, Venue management, Merchandise management, Reports management and Announcement management. High-level conceptual modules structure is shown on the pic. 3. Description of their functions and corresponding user groups, who have access to these functions is described in table 1.



Pic. 3. Modules of the system

Table 1. Modules functions and corresponding user groups

Module	**Function**	**User group**
User Management	Registration	Unregistered user
	Authentication	Guest, Regular user, Admin

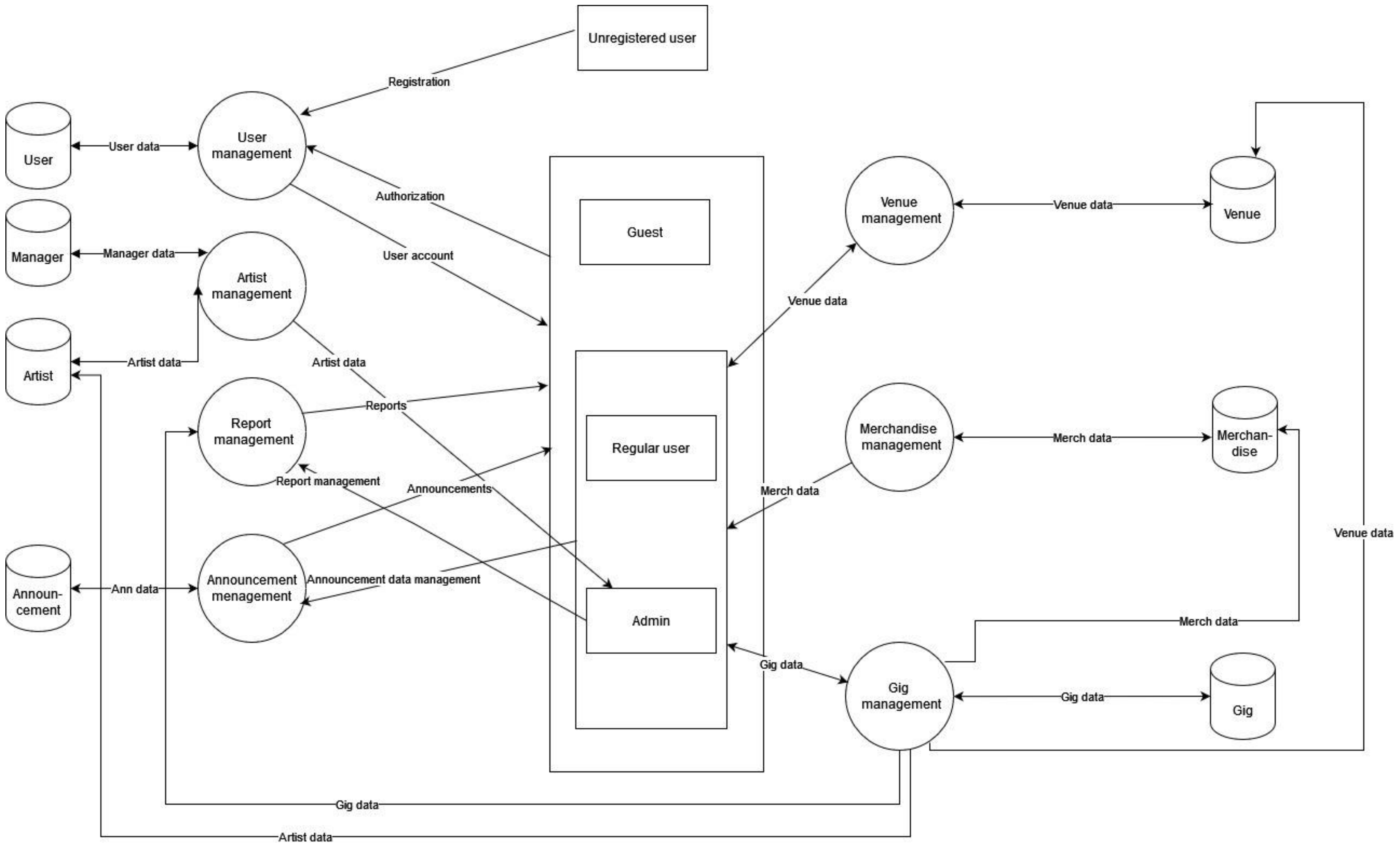
	Logout	Guest, Regular user, Admin
	View user list	Admin
	Create user	Admin
	Delete user	Admin
	Change user role	Admin
Artist Management	Create new artist	Regular user, admin
	View artist	Regular user, admin
	Update artist	Regular user, admin
	Delete artist	Admin
	View artist list	Regular user, Admin
	View gig list of an artist	Regular user, admin
Gig management	Register new gig	Regular user, admin
	Update non-financial information of the gig	Regular user, admin
	Update fin. information of gig	Admin
	View gig list	Regular user, admin
	Add gig template	Admin
	Update gig template	Admin
	Delete gig template	Admin
Venue management	Add new venue	Regular user, admin
	Update venue	Regular user, admin
	View venue list	Regular user, admin
Merchandise management	Add merchandise to the list	Admin
	Add sold merchandise to gig	Regular user, Admin
Reports management	Add a new report	Regular user, Admin
	View the report	Guest, Regular user, Admin

Announcement management	View public announcements	Guest, Regular user, Admin
	Create announcement	Admin
	Delete announcement	Admin
	Edit all announcement	Admin
	Receive Receive email notification	Guest, User, Admin
	Disable/enable email notification	Guest, User

2.2.2. Functional breakdown by modules

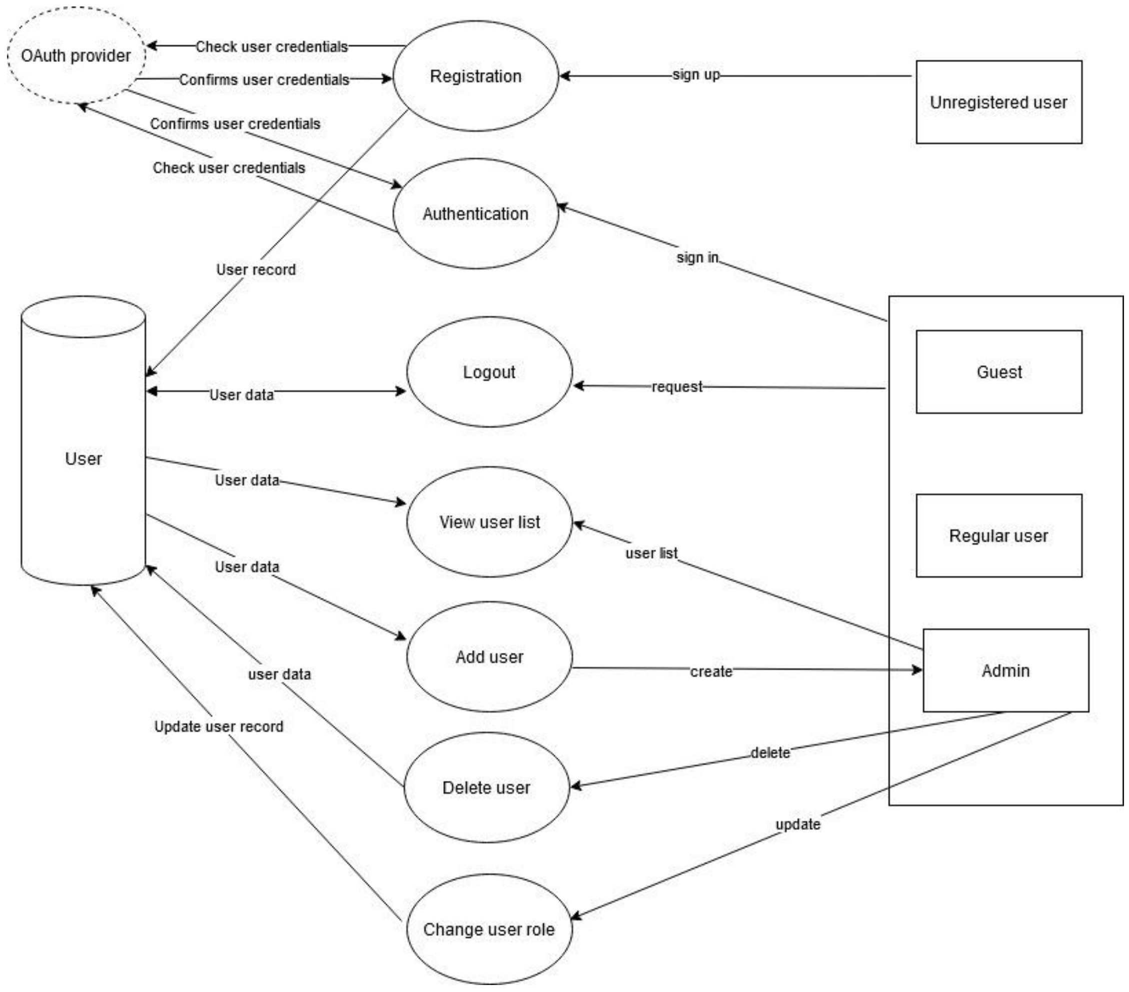
The data flow diagram level 1 (pic. 4), showing functional breakdown is based on the description of the functions in the table above, divided in modules. Several modules are interconnected with each other and require data from another module to be able to process the request. These modules are:

- User management module requires data about linked artists to reflect data in the user list and about the announcements set for each user.
- Artist management module requires data about users and data about gigs registered for the artist.
- Gig management module requires data about artists, venues, merchandise modules etc.
- Venue management module provides data regarding venues, locations and backlines in the venues.
- Merchandise management module provides data about merchandise - list and sold merchandise.
- Report management module requires data about gigs, venues, tickets, locations
- Announcement module requires data from Reports module and Gig module to send announcements



Pic. 4. Data flow diagram (level 1)

2.2.3. User management



Pic. 5. User management module – data flow diagram (level 2)

Table 2. User management module – registration

ID	**User Group**
REGISTRATION	UNREGISTERED USER
NAME	
Registration through OAuth authentication flow	
Description	
System should register a user account by checking user identity through OAuth protocol via provider (Google, Facebook).	

Pre-conditions
User should have Google or Facebook account User has not already been registered to application
Input
1. Name – up to 255 characters 2. E-mail – up to 255 characters
Processing
1. OAuth flow 2. Displays input field for User's profile information - name and email: a. Checks if the email has '@' and '.' in it. If not, then error message (1) b. Checks the length of email. If does not conform, then error message (2) c. Checks the length of the user's name. If does not conform, then error message (3) 3. User data are saved in the database table
Output
If the user is successfully added to the system, he is logged in and message "Welcome, <user> to Gig Story!" appears
Post-conditions
If a user is successfully added to the system, he will be logged in the system.
Error messages
(1) Please include '@' and '.' in the email address. (2) Name cannot be greater than 255 characters. (3) Email cannot be less than 5 and greater than 255 characters.

Table 3. User management module – authentication

ID	**User Group**
AUTHENTICATION	GUEST, REGULAR USER, ADMIN
NAME	
Authentication through OAuth authentication and authorization flow	
Description	
System should authenticate a user by checking user identity through OAuth protocol via provider	

(Google, Facebook) and authorize by checking its role.
Pre-conditions
1. User should have Google or Facebook account 2. User should already be registered to application 3. User is not logged in the application
Input
1. Selection of OAuth provider 2. Remember me (optional)
Processing
1. OAuth authorization flow 2. If not successful, error message (1)
Output
If the user is successfully logged in, message "Welcome, <user> to Gig Story!" appears
Post-conditions
User is successfully logged in the application
Error messages
(1) User is not registered in the system

Table 4. User management module – create a new user

ID	**User Group**
CRT USER	ADMIN
NAME	
Create a new user	
Description	
System should allow admin to create a new user	
Pre-conditions	
User is logged in the system as admin	

Input
3. Name – up to 255 characters 4. E-mail – up to 255 characters
Processing
1. Displays input field for User's profile information - name and email: a. Checks if the email has '@' and '.' in it. If not, then error message (1) b. Checks the length of the email. If does not conform, then error message (2) c. Checks the length of the user's name. If does not conform, then error message (3) 2. Adds a new user record with inserted data to the database. 3. User data are saved in the database table
Output
Message "User <name> saved in the database!
Post-conditions
If a user is successfully created in the system, he will receive an email notification to log in the system.
Error messages
(1) Please include '@' and '.' in the email address. (2) Name cannot be greater than 255 characters. (3) Email cannot be less than 5 and greater than 255 characters.

Table 5. User management module – logout

ID	**User Group**
LOGOUT	GUEST, REGULAR USER, ADMIN
NAME	
Logout	
Description	
Logout of the system by any registered user	
Pre-conditions	
User is logged in the system	

Processing
Logs out from the system by manipulations with the session
Post-conditions
User is logged out of the system
Output
Main page of the system

Table 6. User management module – view user list

ID	**User Group**
USR LIST	ADMIN
NAME	
View user list	
Description	
Processes and shows the list of all users created by the admin with corresponding user role	
Pre-conditions	
User is logged in the system as admin	
Processing	
1. Selects all users from the database (name, email, role) registered by the admin 2. Adds Change user role (table 7) function to each user	
Output	
Returns a formatted list of all users created by the admin, consisting of full name, email, role, and change role function (table 7). At least there always will be 1 user – admin itself.	

Table 7. User management module – change user role

ID	**User Group**
CHG USER ROLE	ADMIN

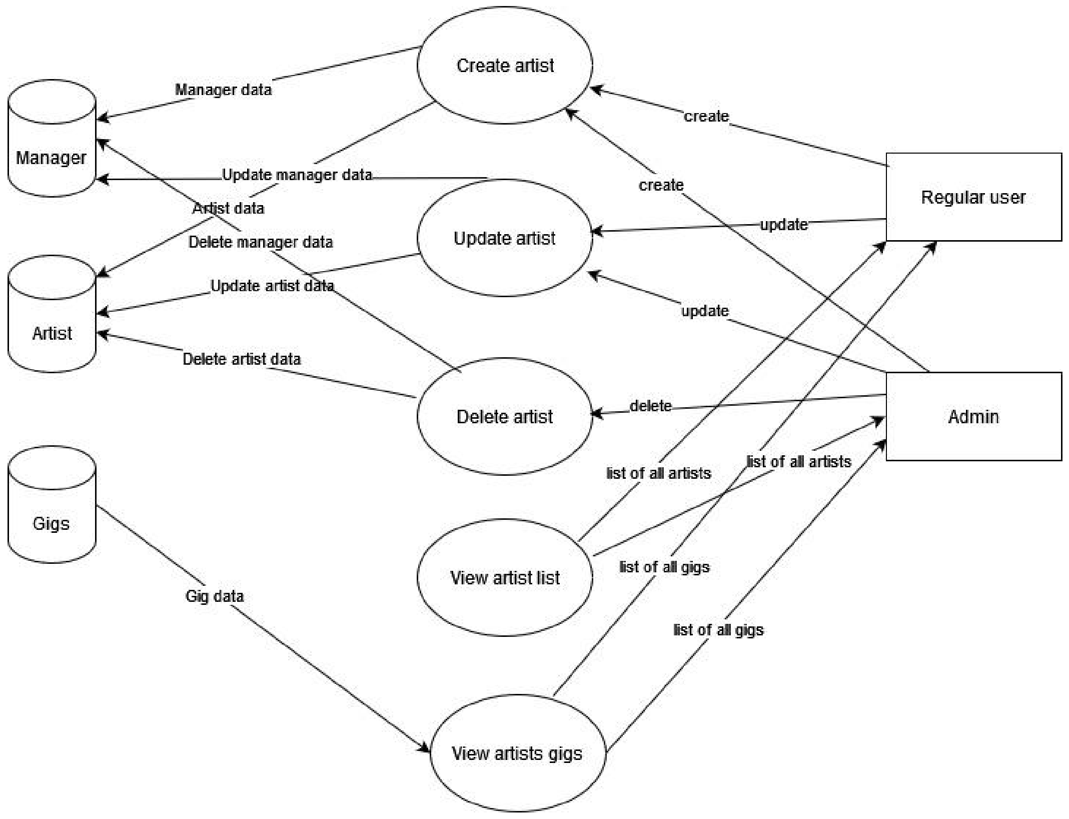
NAME
Change user role
Description
Changes the role of the selected user to different to change the user's rights
Pre-conditions
1. User is logged to the system as Admin 2. Opened list of all users by View user list function (table 6) 3. Select function "change user role" for a specific use
Processing
1. Updates record field "user role" of the selected user in the database based on the new role chosen from the drop-down list.
Output
Returns a formatted list of all users of the system (by View user list function, table 6) with changed user role.

Table 8. User management module – delete a user

ID	**User Group**
DLT USER	ADMIN
NAME	
Delete a user	
Description	
Deletes a user, having role "Regular user" or "Guest", created by admin, from the system	
Pre-conditions	
User is logged in the system as admin	
Input	
Select user to delete	
Processing	

1. Finds user in the database 2. Deletes user record from the database
Output
Message “<User> deleted!” Returns a formatted list of all users of the system (by View user list function, table 6)

2.2.4. Artist management



Pic. 6. Artist management module – data flow diagram (level 2)

Table 9. Artist management module – create a new artist

ID	**User Group**
CRT ARTIST	REGULAR USER, ADMIN
NAME	
Create a new artist profile	
Description	

System should allow regular and admin users to create a new artist profile
Pre-conditions
1. User is registered and logged into system 2. User has role “regular user” or “admin’
Input
1. Name – up to 100 characters 2. Type – from drop-down list 3. E-mail – up to 255 characters 4. Manager’s name – up to 100 characters 5. Manager’s email – up to 255 characters 6. Manager’s phone number – up to 25 characters
Processing
1. Displays input field for Artist’s profile information - name, email, type, manager: a. Checks if the email has '@' and '.' in it. If not, then error message (1) b. Checks the length of the email. If does not conform, then error message (2) c. Checks the length of the artist's name. If does not conform, then error message (3) d. Checks the length of the manager's name. If does not conform, then error message (3) e. Checks the length of the manager's email. If does not conform, then error message (2) f. Checks if the manager’s email has '@' and '.' in it. If not, then error message (1) 2. Adds new artist record with inserted data to the database. 3. Adds new manager record with inserted data to the database (if manager not yet present)
Output
If artist profile data entered correctly, display message: “Artist <name> has been added!
Error messages
(1) Please include '@' and '.' in the email address. (2) Email cannot be less than 5 and greater than 255 characters. (3) Name cannot be greater than 100 characters.

Table 10. Artist management module – update an artist

ID	**User Group**
UPD ARTIST	REGULAR USER, ADMIN
NAME	

Update an artist profile
Description
System should allow regular and admin users to update information in the artist profile
Pre-conditions
1. User is registered and logged into system 2. User has role “regular user” or “admin’ 3. Open artist list by View artist list function (table 11)
Input
All fields are pre-filled with database data.. Data can be changed in fields: ● Name – up to 100 characters ● Type – from drop-down list ● E-mail – up to 255 characters ● Manager’s name – up to 100 characters ● Manager’s email – up to 255 characters ● Manager’s phone number – up to 25 characters
Processing
1. Fields data validation. a. Checks if the email has '@' and '.' in it. If not, then error message (1) b. Checks the length of email. If does not conform, then error message (2) c. Checks the length of the artist's name. If does not conform, then error message (3) d. Checks the length of the manager's name. If does not conform, then error message (3) e. Checks the length of the manager's email. If does not conform, then error message (2) f. Checks if the manager's email has '@' and '.' in it. If not, then error message (1) 2. Updates record fields of selected artist if changed 3. Updates record fields of corresponding manager data if changed
Output
Returns a formatted data about all artists (table 11) created by the user with changed data
Error messages
(1) Please include '@' and '.' in the email address. (2) Email cannot be less than 5 and greater than 255 characters. (3) Name cannot be greater than 100 characters.

Table 11. Artist management module – view artist list

ID	**User Group**
VIEW ARTIST LIST	REGULAR USER, ADMIN
NAME	
View all artists	
Description	
System should allow regular users and admin users to view list of all artists	
Pre-conditions	
1. User is registered and logged into system 2. User has role “regular user” or “admin’	
Processing	
1. Selects all artists from the database (name, email, type, manager) 2. Adds Update artist function (table 10) to each artist 3. If user does not have an artist entity registered, error message (1)	
Output	
Returns a formatted list of all artists for the regular user, consisting of name, email, type and manager, and Update artist function (table 10).	
Error messages	
(1) You do not have any artists registered	

Table 12. Artist management module – delete an artist

ID	**User Group**
DLT ARTIST	ADMIN
NAME	
Delete an artist profile from the database of the system	
Description	
System should allow admin users to delete an artist profile	

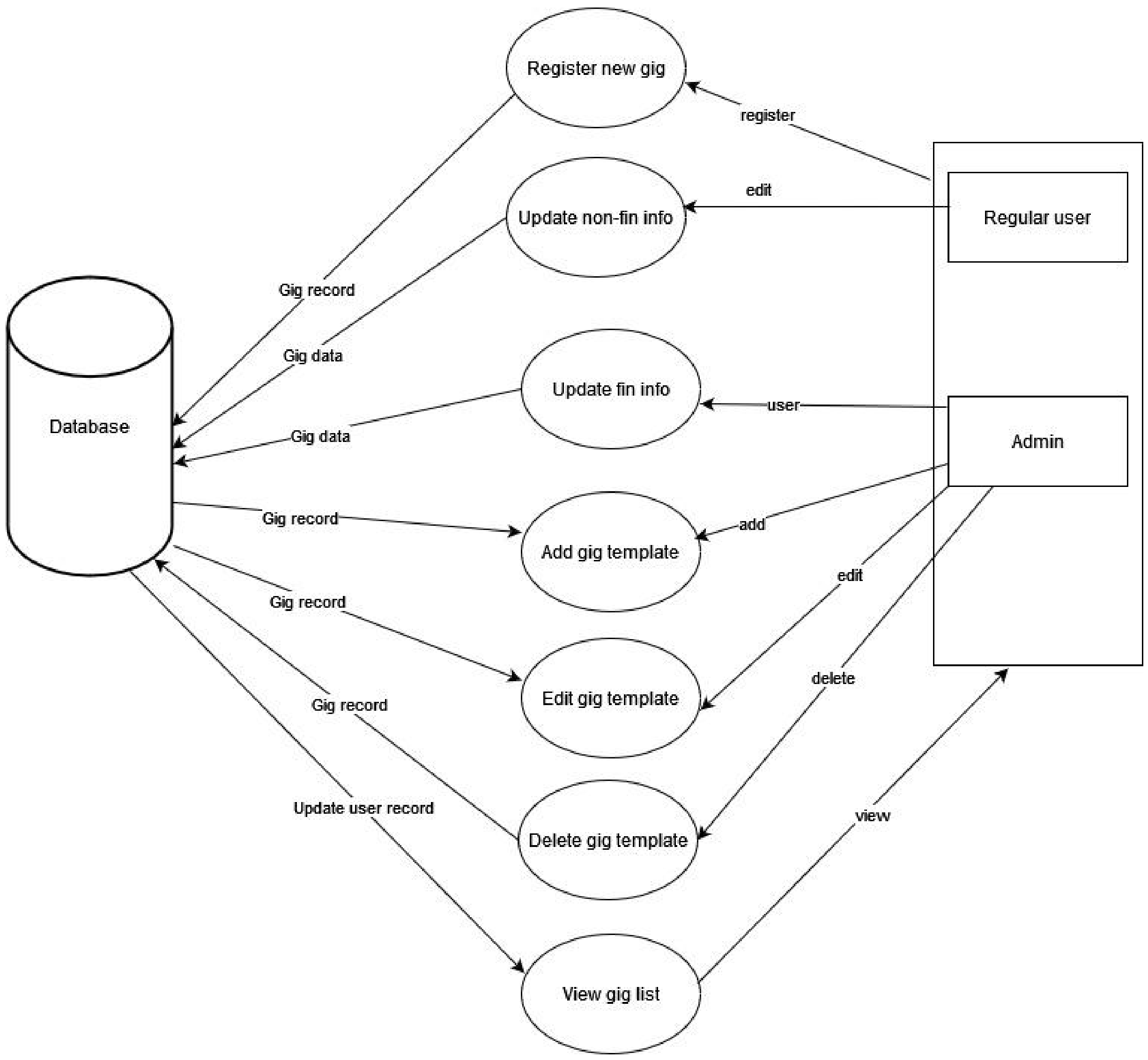
Pre-conditions
1. User is registered and logged into system 2. User has role “admin’ 4. User has opened “view artist list” (table 11) 3. Artist does not have any gigs registered
Input
Select artist to delete
Processing
1. Finds artist in the database 2. Checks if artist has any registered gigs a. if yes, displays error message (1) b. if no, deletes artist record from the database
Output
Returns artist list with made changes (table 11)
Error messages
(1) Artist <name> has registered gigs and cannot be deleted

Table 13. Artist management module – View gig list of an artist

ID	**User Group**
VIEW ART GIGS	REGULAR USER, ADMIN
NAME	
View gig list of an artist	
Description	
System must allow regular and admin users to view a list of gigs for an artist	
Pre-conditions	
1. User is registered and logged into system 2. User has role “regular user” or “admin’	
Input	

Select artist to see list of gigs
Processing
1. Checks if artist has registered gigs: a. if any gigs registered for the artist, then error message (1) b. if gigs found – selects necessary fields from the database for all found gigs
Output
Returns formatted data about all gigs registered for the artist
Error messages
(1) Artist <name> does not have any registered gigs yet.

2.2.5. Gig management



Pic. 7. Gig management module – data flow diagram (level 2)

Table 14. Gig management module – register a new gig

ID	User Group
REG GIG	REGULAR USER, ADMIN
NAME	
Register a new gig	

Description

System should allow regular and admin users to register a new gig

Pre-conditions

1. User is registered and logged into system
2. User has role “regular user” or “admin’

Input

1. Name – up to 50 characters, optional
2. Artist - from dropdown list, mandatory
3. Venue - from dropdown list, mandatory
4. Location - from dropdown list, mandatory
5. Date - in YYYY-MM-DD format, mandatory if isTemplate = false, not available, if isTemplate = true
6. Day of week - from dropdown list, mandatory if isTemplate = false, not available, if isTemplate = true
7. Starting time - in HH:MM, mandatory if isTemplate = false, not available, if isTemplate = true
8. Type – from dropdown list, mandatory
9. Ticket – price (int), quantity (int), optional
5. Location – country from dropdown list, region from dropdown lists, city from dropdown lists, mandatory in case if isOnline = false
6. Venue - type from dropdown list, name up to 100 characters, capacity (number), notes up to 300 characters, mandatory in case if isOnline = false
7. Backline - description up to 255 characters, costs, optional
8. Attendees - number, average age range, description up to 255 characters, mandatory
9. Merchandise sold - type from dropdown list and quantity, optional
10. Guests - number, optional
11. Media Representatives - name up to 100 characters, media name up to 100 characters, media type from dropdown list, optional
12. is festival - boolean, default = false
13. is online - boolean, default = false
14. is tour - boolean, default = false
15. is template - boolean, default = false
16. Notes - up to 255 characters, may be multiple notes, optional

Processing

1. Displays choice to add new gig from template:
    a. if user selects yes, display template fields from gig records, which are isTemplate = true in the database;
    b. if user selects no, display all fields:
        i. Checks the length of gig name. If does not conform, then error message (1)
        ii. Checks the date format. If does not conform, then error message (2)

iii. Checks the time format. If does not conform, then error message (3) iv. Checks media representative and venue name length. If does not conform, then error message (4) v. Checks notes and description field length. If does not conform, then error message (5) vi. Checks if mandatory fields have been filled in. If not, error message (6) 2. Adds a new gig record with inserted data to the database, in the respective entities where data entered.
Output
"You just registered a new gig!"
Error messages
(1) Gig name cannot be shorter than 5 and greater than 50 characters. (2) Date should be in format YYYY-MM-DD (3) Time should be in format HH:MM (4) This field cannot be shorter than 5 and greater than 100 characters. (5) This field cannot be shorter than 5 and greater than 300 characters. (6) Please fill in all mandatory fields (with *).

Table 15. Gig management module – add financial information of a gig

ID	**User Group**
UPD GIG WITH FIN INFO	ADMIN
NAME	
Add financial information to the gig	
Description	
System should allow admin users to add financial information to a registered gig	
Pre-conditions	
1. User is registered and logged into system 2. User has role "admin" 3. Open gig list by View list of gigs function (table 20)	
Input	

1. All fields are pre-filled with database data 2. Data can be added / updated in fields: a. Fee type from dropdown list, amount, taxesPaid, optional b. Merchandise income, mandatory in case if merchandise quantity registered c. Production costs, optional
Processing
1. Fields data validation: a. checks if the merchandise income field is filled in case the field “merchandise quantity” contains value. If not, then error message (1) 2. Update gig record with inserted data to the database in the respective tables.
Output
“Financial data added for the gig!”
Error messages
(1) This gig has registered sold merchandise, please fill in merch income field

Table 16. Gig management module – edit non-financial information of a saved gig

ID	**User Group**
UPDATE NON-FIN INFO GIG	REGULAR USER, ADMIN
NAME	
Edit non-financial information of a gig	
Description	
System should allow regular and admin users to edit any non-financial information of a registered gig	
Pre-conditions	
1. User is registered and logged into system 2. User has role “regular user” or “admin” 3. Open gig list by View list of gigs function (table 20)	
Input	
1. All fields are pre-filled with database data 2. Data can be changed in fields, except Fee, Merchandise income, Productions costs	

Processing
1. Field data verification: a. Checks length of gig name. If does not conform, then error message (1) b. Checks date format. If does not conform, then error message (2) c. Checks time format. If does not conform, then error message (3) d. Checks media representative and venue name length. If does not conform, then error message (4) e. Checks notes and description field length. If does not conform, then error message (5) f. Checks if mandatory fields have been filled in. If not, error message (6) 2. Updates gig record with data from fields to the database in the respective tables.
Output
"Information updated for the gig!"
Error messages
(1) Gig name cannot be shorter than 5 and greater than 50 characters. (2) Date should be in format YYYY-MM-DD (3) Time should be in format HH:MM (4) This field cannot be shorter than 5 and greater than 100 characters. (5) This field cannot be shorter than 5 and greater than 300 characters. (6) Please fill in all mandatory fields (with *).

Table 17. Gig management module – add gig template

ID	**User Group**
ADD GIG TEMPLATE	ADMIN
NAME	
System should allow admin users to add gig records with attribute "isTemplate" = true	
Description	
System should allow admin users to edit the financial information of a registered gig	
Pre-conditions	
1. User is registered and logged into system 2. User has role "admin'	

Input

1. Name – up to 50 characters, mandatory if isTemplate = true
2. Date – in YYYY-MM-DD format, not available for templates
3. Day of week - not available for templates
17. Starting time - in HH:MM, not available for templates
18. Type – from dropdown list, mandatory
19. Ticket – price (int), quantity (int), optional
20. Location – country from dropdown list, region from dropdown lists, city from dropdown lists, optional if isTemplate = true
21. Venue - type from dropdown list, name up to 100 characters, capacity (number), notes up to 255 characters, optional if isTemplate = true
22. Backline - description up to 300 characters, costs, optional
23. Attendees - number, average age range, description up to 255 characters, mandatory
24. Merchandise sold - type from dropdown list and quantity, optional
25. Guests - number, optional
26. Media Representatives - name up to 100 characters, media name up to 100 characters, media type from dropdown list, optional
27. is festival - boolean, default = false
28. is online - boolean, default = false
29. is tour - boolean, default = false
30. is template - boolean, mandatory = true
31. Notes - up to 255 characters, may be multiple notes, optional

Processing

1. Field validation:
    a. Checks length of template name. If does not conform, then error message (1)
    b. Checks media representative and venue name length. If does not conform, then error message (2)
    c. Checks notes and description field length. If does not conform, then error message (3)
    d. Checks if mandatory fields have been filled in. If not, error message (4)

2. Adds a new gig record as a template with inserted data to the database.

Output

“You added a new template with name <name>”

Error messages

(1) Template name cannot be shorter than 5 and greater than 50 characters.
(2) This field cannot be shorter than 5 and greater than 100 characters.
(3) This field cannot be shorter than 5 and greater than 300 characters.
(4) Please fill in all mandatory fields (with *).

Table 18. Gig management module – edit gig template

ID	**User Group**
EDIT GIG TEMPLATE	ADMIN
NAME	
Edit template of a gig	
Description	
System should allow admin users to edit the gig records saved with “isTemplate” = true	
Pre-conditions	
1. User is registered and logged into system 2. User has role “admin’ 3. Open gig list by View gig list function, by choosing filter for templates (table 20)	
Input	
1. All fields are pre-filled with database data 2. Data can be changed in all available fields	
Processing	
1. Field data verification: g. Checks the length of the template name. If does not conform, then error message (1) h. Checks media representative and venue name length. If does not conform, then error message (2) i. Checks notes and description field length. If does not conform, error message (3) j. Checks if mandatory fields have been filled in. If not, error message (4) 2. Updates gig as a template record with data from fields to the database.	
Output	
“You have updated a template with name <name>!”	
Error messages	
(1) Template name cannot be shorter than 5 and greater than 50 characters. (2) This field cannot be shorter than 5 and greater than 100 characters. (3) This field cannot be shorter than 5 and greater than 300 characters. (4) Please fill in all mandatory fields (with *).	

Table 19. Gig management module – delete gig template

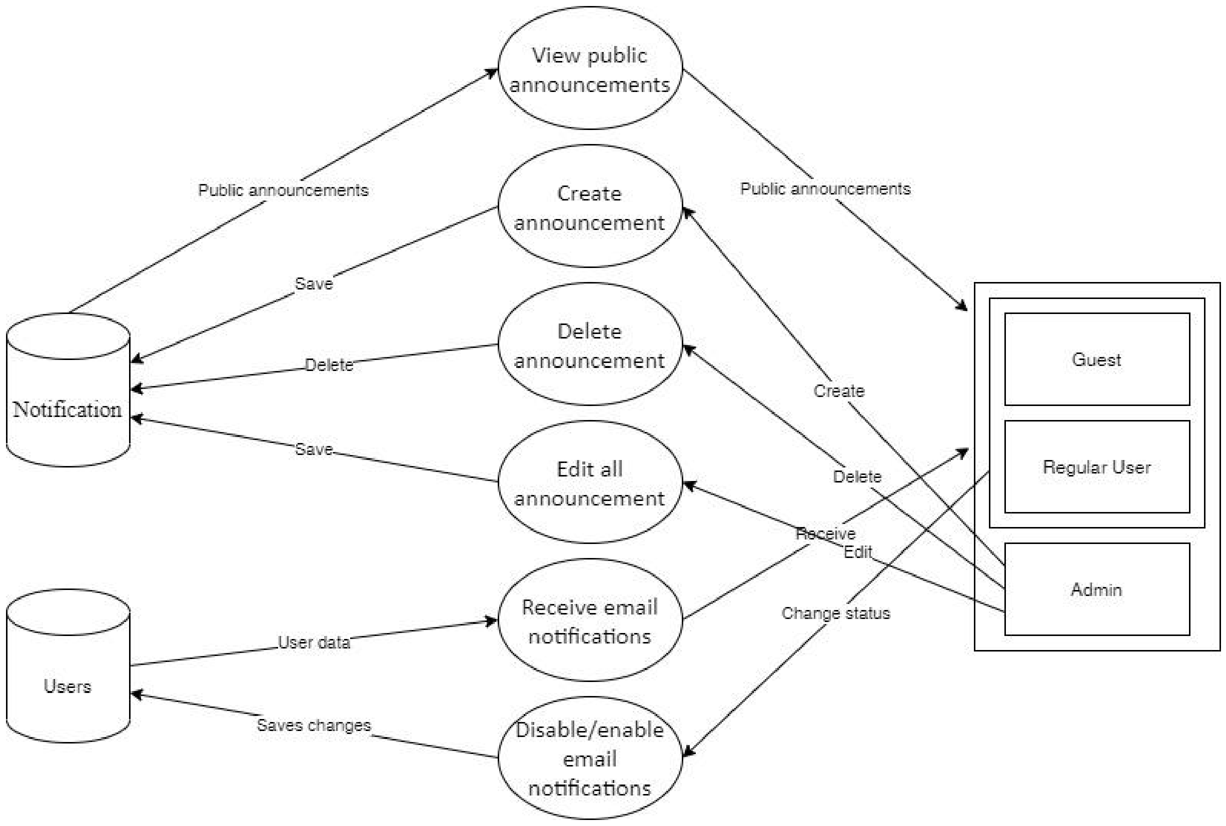
ID	**User Group**
DELETE GIG TEMPLATE	ADMIN
NAME	
Delete gig template	
Description	
System should allow admin users to delete the gig records saved with “isTemplate” = true	
Pre-conditions	
1. User is registered and logged into system 2. User has role “admin” 3. Open gig list by View gig list function, by choosing filter for templates (table 20)	
Input	
Select template to delete	
Processing	
1. Finds gig as a template record in the database 2. Deletes gig record from the database	
Output	
Returns list of gigs with made changes (table 20)	

Table 20. Gig management module – View list of gigs

ID	**User Group**
VIEW GIG LIST	REGULAR USER, ADMIN
NAME	
View gig list	
Description	
System must allow regular and admin users to view a list of registered gigs	

Pre-conditions
1. User is registered and logged into system 2. User has role “regular user” or “admin’
Input
Select gig list from a filter (by type, date, period, starting time, artist, location, venue)
Processing
1. Checks if any registered gigs for the selected filter criteria a. if no gigs found, then error message (1) b. if gigs found – selects necessary fields from the database for all found gigs
Output
Returns formatted data about all gigs registered corresponding to the filter criteria.
Error messages
(1) No gigs corresponding to the selected filter criteria registered

2.2.6. Announcement management



Pic. 8. Announcement management module – data flow diagram (level 2)

Table 21. Announcement module – View public announcements

ID	**User Group**
PBL ANNOUNCEMENTS	GUEST, REGULAR USER, ADMIN
NAME	
View public announcements	
Description	
System must allow all logged in users to view public announcements	
Pre-conditions	
User is registered and logged into system	
Processing	

Selects and displays all public announcements from the database
Output
Shows all public announcements in a preformatted way (basically forming the main page for Guest, Regular user and Admin.

Table 22. Announcement module – Create announcement

ID	**User Group**
CRT ANNOUNCEMENT	ADMIN
NAME	
Create announcement	
Description	
Allows to see create new announcements in the system	
Pre-conditions	
1. Logged to the system as Admin 2. Open a list of all announcements 3. Select "create new announcement"	
Input	
• Header text • Message text • Announcement type • Announcement language	
Processing	
1. Validates all fields a. if filled left blank, then error (1) 2. Adds new announcement to the database	
Output	
Returns to the main page of application and shows added announcement	
Error messages	
(1) All fields have to be filled	

Table 23. Announcement module – Delete announcement

ID	**User Group**
DLT ANNOUNCEMENT	ADMIN
NAME	
Delete announcement	
Description	
Allows to delete any announcement in the system	
Pre-conditions	
1. Logged to the system as Admin 2. Open a list of all announcements	
Input	
Select announcement to delete	
Processing	
Deletes selected announcement from the database	
Output	
Returns to the main page of application and shows all the active announcements except the deleted one	

Table 24. Announcement module – Edit announcement

ID	**User Group**
EDT ANNOUNCEMENT	ADMIN
NAME	
Edit announcement	
Description	
Allows to edit all announcements in the system	
Pre-conditions	

1. Logged to the system as Admin 2. Open a list of all announcements
Input
• Select announcement to edit • Change data of fields (header text, message text, announcement type)
Processing
1. Validates all fields a. if field left blank, then error (1) 2. Updates selected announcement from the database
Output
Returns to the main page of application and shows edited notification and all active announcements
Error messages
(1) All fields have to be filled

Table 25. Announcement module – Receive email notification

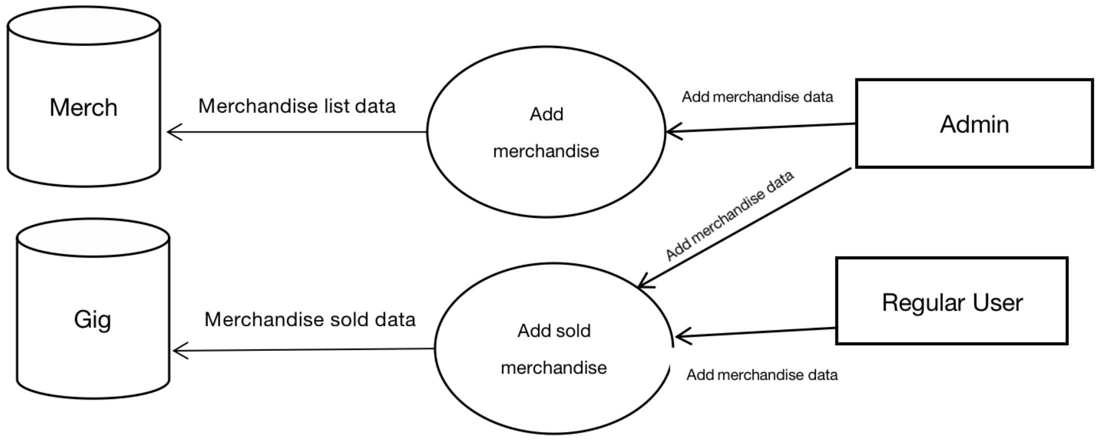
ID	**User Group**
RCV EMAIL NTF	GUEST, REGULAR USER, ADMIN
NAME	
Receive email notification	
Description	
Sends email notification based on contents got from the new announcements	
Pre-conditions	
1. Be Guest, Regular user, Admin 2. Has email notifications turned on (table 26)	
Input	
Data for message from new announcements data	
Processing	

1. Checks if email notifications turned on a. if no, then terminates 2. Composes email based on the necessary information 3. Sends email to corresponding user email address
Output
Sends email to the user email address

Table 26. Announcement module – Disable/enable email notification

ID	**User Group**
ENBL EMAIL NTF	GUEST, REGULAR USER
NAME	
Disable/enable email notification	
Description	
Allows to turn off and on email notification	
Pre-conditions	
1. Logged to the system as Guest, User 2. Opened “View user list function (table 6)	
Input	
Select email notification status (on/off)	
Processing	
Updates data in the database according to selected option	
Output	
Success message on the same page	

2.2.7. Merchandise management



Pic. 9. Merchandise management module – data flow diagram (level 2)

Table 27. Merchandise management module – add merchandise item to the list

ID	User Group
ADD MERCH	ADMIN
NAME	
Add merchandise item to the list	
Description	
User with admin role can add merchandise	
Pre-conditions	
1. User is registered and logged into system 2. User has role "Admin"	
Input	
1. Merchandise name - up to 255 characters, mandatory 2. Merchandise type - from dropdown list, mandatory 3. Quantity, optional 4. Price, optional	

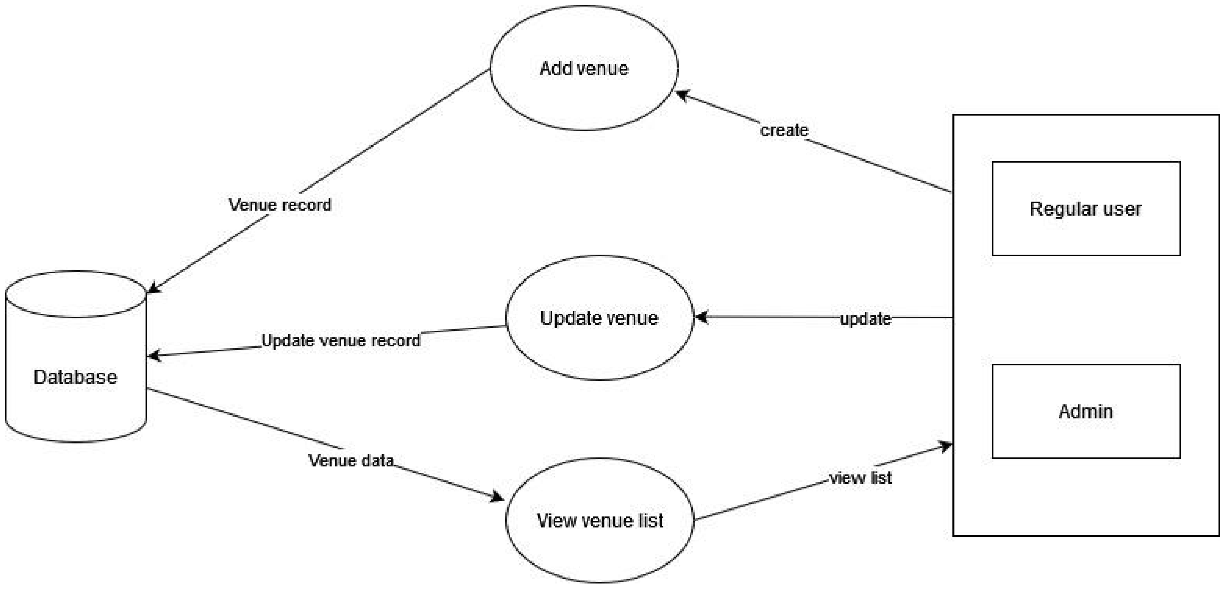
Processing
1. Displays input field for Merchandise item information - name, type, price, quantity   a. Checks length of item name. If does not conform, then error message (1)   b. Checks if mandatory fields are filled in. If not, then error message (2) 2. Adds new merchandise record with inserted data to the database.
Output
"Merchandise <name> has been added!
Error messages
(1) Item name cannot be greater than 255 characters. (2) Please fill in all mandatory fields (with *).

Table 28. Merchandise management module – add sold merchandise to the gig

ID	**User Group**
ADD SOLD MERCH	REGULAR USER, ADMIN
NAME	
Add sold merchandise to the gig	
Description	
System should allow users to add sold merchandise to an already registered gig	
Pre-conditions	
1. User is registered and logged into system 2. User has role "regular user" or "admin"	
Input	
1. Merchandise name - dropdown from merchandise list, mandatory 2. Quantity, mandatory 3. Gig - dropdown, mandatory	
Processing	
1. Displays input field for Merchandise item information - name, quantity	

a. Checks if mandatory fields are filled in. If not, then error message (1) 2. Adds new sold merchandise record with inserted data to the database.
Output
“You have added <merchandise> sold in the gig.”
Error messages
(1) Please fill in all mandatory fields (with *).

2.2.8. Venue management



Pic. 10. Venue management module – data flow diagram (level 2)

Table 29. Venue management module – add a new venue

ID	**User Group**
ADD VENUE	REGULAR USER, ADMIN
NAME	
Add a new venue	

Description
User can add a new venue
Pre-conditions
1. User is registered and logged into system 2. User has role “regular user” or “admin”
Input
1. Venue name - up to 100 characters, mandatory 2. Venue type - from dropdown, mandatory 3. Venue capacity, optional 4. Location - city, region, country from dropdown lists, mandatory 5. Backline costs - int 6. Backline description - up to 255 characters, optional
Processing
1. Displays input field for Venue information - name, type, capacity, location, backline. a. Checks length of name. If does not conform, then error message (1) b. Checks length of backline description. If does not conform, then error message (2)
Output
“Venue <name> added!”
Error messages
(1) Name cannot be greater than 100 characters. (2) Backline description cannot be greater than 255 characters. (3) Please fill in all mandatory fields (with *)

Table 30. Venue management module – Update a venue

ID	**User Group**
UPD VENUE	REGULAR USER, ADMIN
NAME	
View a venue name	

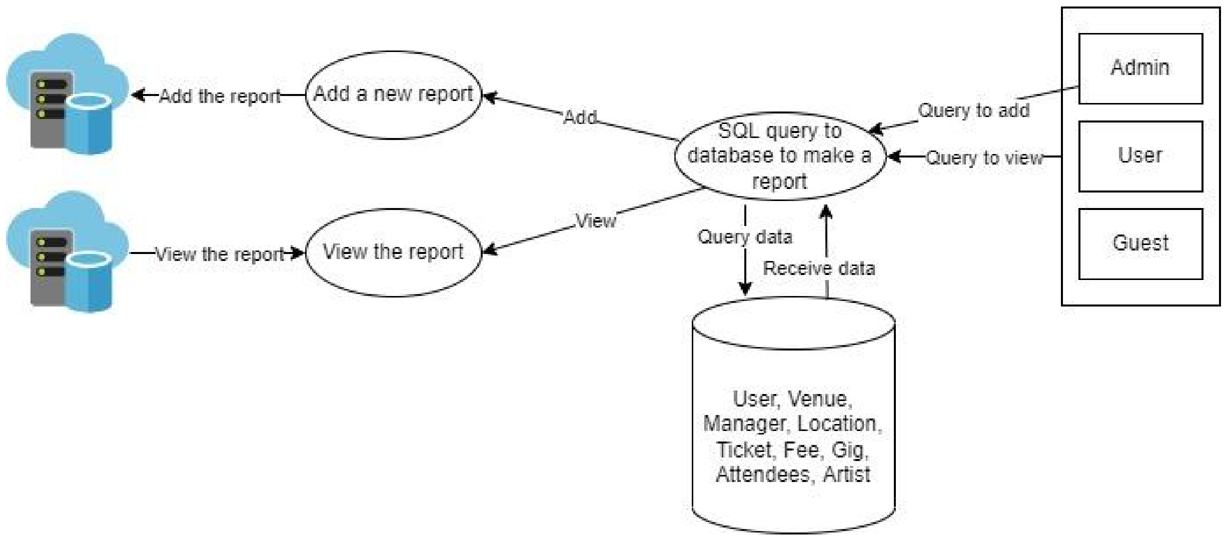
Description
System should allow regular and admin users to update venue information
Pre-conditions
1. User is registered and logged into system 2. User has role “regular user” or “admin” 3. Open artist list by View venues function (table 31)
Input
All fields are pre-filled with database data Data can be changed in fields: ● Name - up to 255 characters, mandatory ● Type, mandatory ● Capacity, optional ● Location ● Location - city, region, country from dropdown lists, mandatory ● Backline costs - optional ● Backline description - up to 300 characters, optional
Processing
1. Displays input field for Venue information - name, type, capacity, location, backline. c. Checks length of name. If does not conform, then error message (1) d. Checks length of backline description. If does not conform, then error message (2) e. Checks if all mandatory fields filled. If not, then error message (3) 2. Updates venue record with new data to the database in the respective fields.
Output
“You updated information for venue <name>!”
Error messages
(1) Name cannot be greater than 255 characters. (2) Backline description cannot be greater than 300 characters. (3) Please fill in all mandatory fields (with *)

Table 31. Venue management module – View venue list

ID	User Group
VIEW VENUE LIST	REGULAR USER, ADMIN
NAME	
View venue list	
Description	
System must allow regular and admin users to view a list of venues in the database	
Pre-conditions	
1. User is registered and logged into system 2. User has role “regular user” or “admin’	
Input	
Select venue list from a filter (by location, type, name)	
Processing	
1. Checks if any registered venues for the selected filter criteria a. if no venues found, then error message (1) b. if venues found – selects necessary fields from the database for all found venues	
Output	
Returns formatted data about all venues registered corresponding to the filter criteria.	
Error messages	
(1) No venues corresponding to the selected filter criteria registered	

2.2.9. Reports management

Reports will not be saved in the database, rather a cloud storage solution will be used. The SQL queries for generating the reports from databases will be created automatically using user input on what data do they need in their report. There will be pre-made forms for that.



Pic. 11. Reports management module - data flow diagram (level 2)

Table 34. Reports module - Add a report

ID	**User Group**
ADD REPORT	ADMIN
NAME	
Add a report	
Description	
System must allow user with admin role to add a report by applying filters to the gig list and saving report	
Pre-conditions	
1. User is registered and logged into system 2. User has role “Admin”	
Input	
User inputs the fields in the premade template forms, such as: “Venue”, “Artist”, “Fees”, “Tickets”, “city” etc, depending on the type of report required	
Processing	

The field values are entered into premade SQL query templates and the report is queried from the database and further added to the cloud storage.
Output
Returns formatted report and saves it to cloud storage
Error messages
(1) Wrong data type on input value (2) Value does not exist in database

Table 35. Reports module - view a report

ID	**User Group**
VIEW REPORT	GUEST, REGULAR USER, ADMIN
NAME	
View a report	
Description	
System must allow user to view created reports choosing from the list of existing reports	
Pre-conditions	
User is registered and logged into system	
Input	
User chooses the report from a list of existing reports, keywords can be added to narrow down the list of reports	
Processing	
The chosen report is received from the cloud storage	
Output	
Returns formatted report	
Error messages	

(1) Report does not exist
(2) <user> does not have permission to view this report

2.3. Non-functional requirements

Portability

- GigStory can be launched on both IOS and Android,
- GigStory can be launched on Windows and Mac.

Compatibility

- In order to receive support for The iOS application iPhone devices must be running on IOS 8 or higher,
- In order to receive support for The Android application Android devices must be running on Lollipop 5.1 or higher.

Reliability

- The system should encounter less than 1 bug per 10 hours of use,
- The system cannot encounter more than 30 minutes of downtime per month.

Security

- The applications can prevent cross site scripting attacks,
- The application does not store hard coded sensitive information.

Usability - It must be easy to learn how to use the application. Otherwise we cannot expect customers to use our system.

- It is necessary for the user to succeed creating an account in the first try,
- It is necessary for the user to succeed creating an "artist" in the first try.
- To reduce customer annoyance and need for personal help from support, it is important that customers understand the causes of malfunctions.

Scalability

- The application must be able to support an annual transactional growth rate of 10%, and still meet all defined transactional performance requirements,
- The database must be able to an annual growth rate of 20 %, with no degradation in database performance,
- The application must be able to support a 10 % growth in user concurrency, and still meet all defined transactional performance requirements.

Performance

- Each request should be processed within 10 seconds.
- All user updates should be autosaved within 2 seconds.
- The app should load in 3 seconds when the number of simultaneous users are > 10000

3. Software design description

3.1. Database design

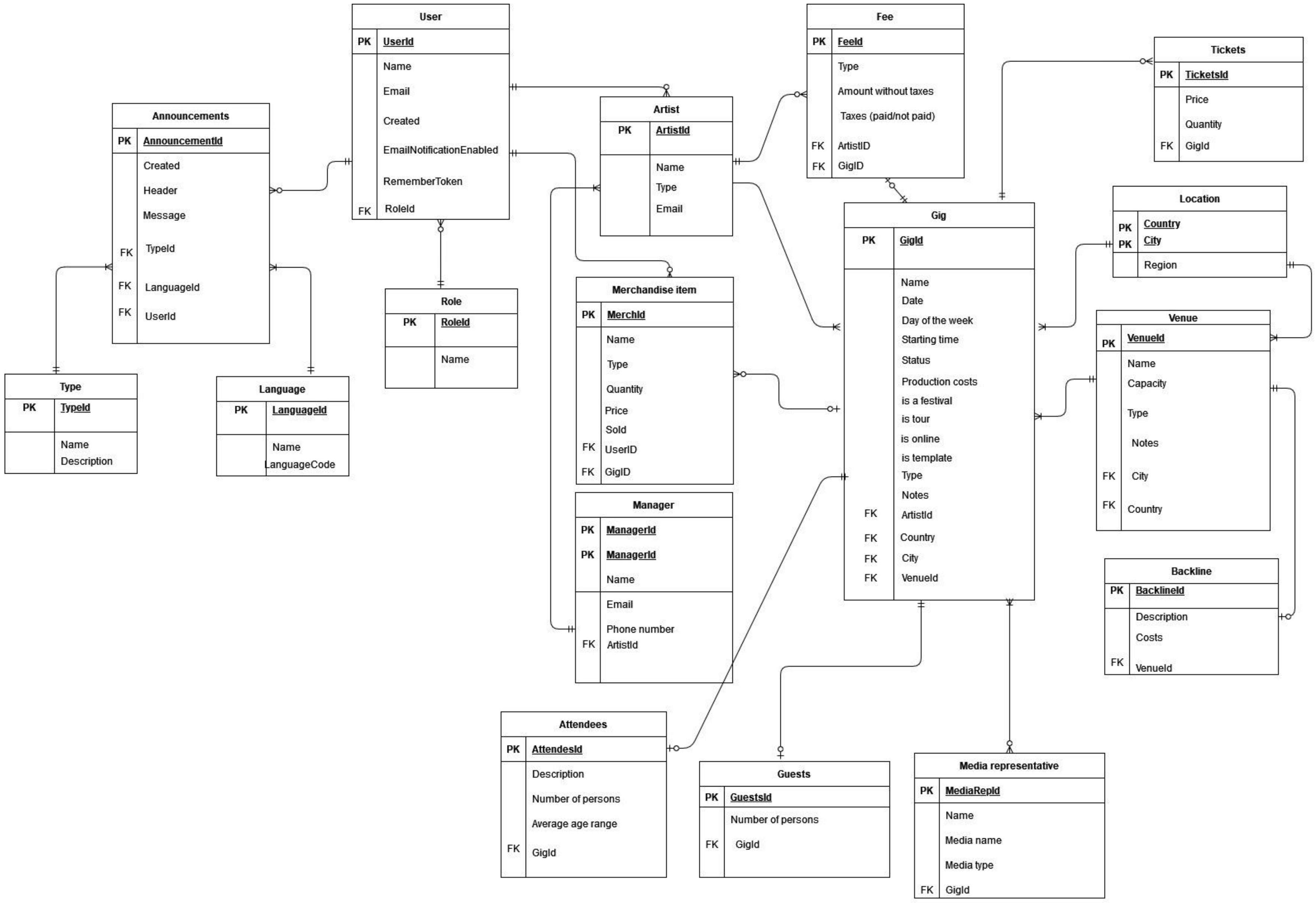
Based on the functional description and conceptual database model (pic. 2) logical database model was created (pic. 12).

The physical database model (pic. 13) was created based on the logical model. It includes one additional entity – conjunction table for N to N relation between MediaRepresentative and Gig entities and shown on the picture as MediaRepInGig entity. All entities include main data types and nullable columns.

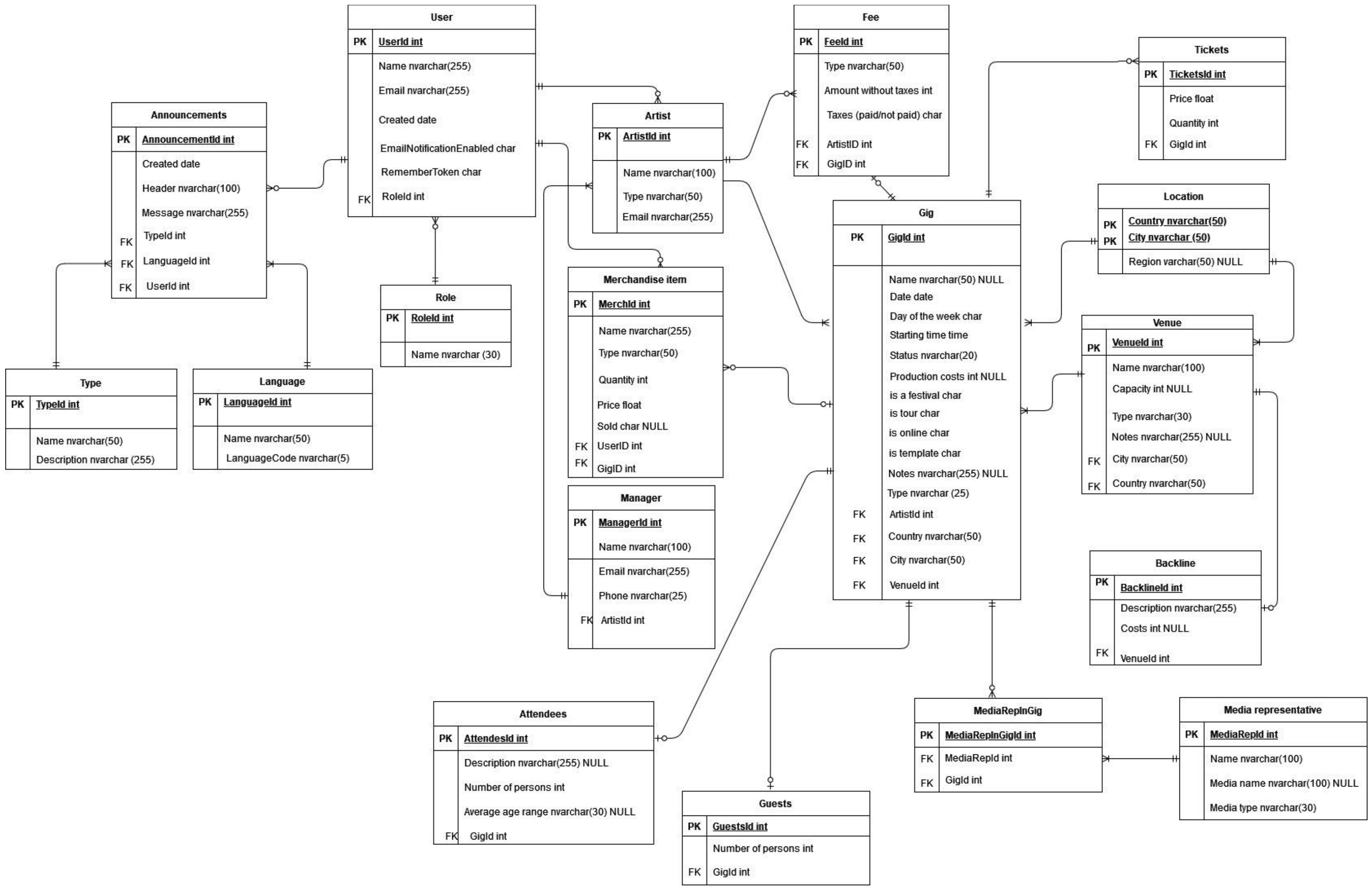
The terms of data types and other applicable parameters to database content may vary depending on the used DBMS. Herein used terms of Microsoft SQL server.

The description of database tables design is split in the subsections according to the modules breakdown shown in the section 2.2.2.

- In subsection 3.1.1. described design of User and Role entities.
- In subsection 3.1.2. described design of Artist and Manager entities
- In subsection 3.1.3. described design of Gig, Fee, Attendees, Guests, MediaRepInGig, MediaRepresentative, Tickets entities
- In subsection 3.1.4. describe design of Venue, Location, Backline entities
- In subsection 3.1.5. describe design of Merchandise entity
- In subsection 3.1.6. described design of Announcements, Type and Languages entities.



Pic. 12. Database logical model



Pic. 13. Database physical model

3.1.1. Users and corresponding entities

This subsection contains detailed information about table design of User entity (table 36) and related entity – Role (table 37).

Data from the User entity is used by User management module functions (section 2.2.3., tables 2-8).

Table 36. USER entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	UserId	int	No	IDENTITY (1, 1)	Automatic id for user entity
	Name	nvarchar(255)	No		Username used in the system
	Email	nvarchar(255)	No	UNIQUE	Unique email address belonging to the user
	Created	date	No		Date of creation of the corresponding user account
	EmailNotificationEnabled	char	No		Contains Y or N value to show whether email notifications turned on or off
	RememberToken	char	No		Contains Y or N value to show whether the remember token has been selected. N by default (if not selected).
FK	RoleId	int	No	FK referenced from entity Role (table 37, id attribute)	The role of the user in the system

User roles are stored in the Role entity table – the roles shown in the section 1.5. of the document.

Table 37. ROLE entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	RoleId	int	No	IDENTITY (1, 1)	Automatic id for user entity
	Name	nvarchar(30)	No		Role name used in the system

3.1.2. Artists and corresponding entities

This subsection contains detailed information about table design of Artist entity (table 38) and related entity – Manager (table 39).

Mainly the data from the Artist entity is used by Artist management module functions (section 2.2.4., tables 9-13) as well as in the Gig management module (table 14-16, 20).

Table 38. ARTIST entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	ArtistId	int	No	IDENTITY (1, 1)	Automatic id for artist entity
	Name	nvarchar(255)	No		Name of the artist
	Type	nvarchar(30)	No		Type of the artist
	Email	nvarchar(255)	No	UNIQUE	Unique email address belonging to the artist

Data from the Manager entity is used in the Artist management module (tables 9-12).

Table 39. MANAGER entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	ManagerId	int	No	IDENTITY (1, 1)	Automatic id for manager entity

	Name	nvarchar(50)	No		Name of the manager
	Phone Number	nvarchar(25)	No		Phone number of the manager
	Email	nvarchar(255)	No	UNIQUE	Unique email address belonging to the manager
FK	ArtistId	int	No	FK referenced from entity Artist (table 38, id attribute)	The artist managed by the manager

3.1.3. Gigs and corresponding entities

This subsection contains detailed information about table design of Gig entity (table 40) and related entities – Fee (table 41), Attendees (table 42), Guests (table 43), MediaRepresentative (table 44), MediaRepInGig (table 45), Tickets (table 46).

Data from the Gig entity is used by Gig Management module functions (section 2.2.5., tables 14-20) as well as in Report management module (section 2.2.9., table 34).

Table 40. GIG entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	GigId	int	No	IDENTITY (1, 1)	Automatic id for gig entity
	Name	nvarchar(50)	Yes		Name of the gig.
	Date	date	No		Date of the gig.
	DayOfTheWeek	char	No		Numbers 1-7 corresponding to days
	Starting time	time	No		Starting time of the gig. No end time or duration will be provided.
	Status	nvarchar(20)	No		Status of the gig (planned, confirmed, completed, canceled)

	isFestival	char	No		Contains Y or N value to show whether the gig is a festival. N by default (if not selected).
	isTour	char	No		Contains Y or N value to show whether the gig is part of a tour. N by default (if not selected).
	isOnline	char	No		Contains Y or N value to show whether the gig is happening online. N by default (if not selected).
	isTemplate	char	No		Contains Y or N value to show whether the gig is saved to database as a template. N by default (if not selected).
	Notes	nvarchar(255)	Yes		Contains body of the notes for the gig.
FK	VenueId	int	No	FK referenced from entity Venue (table 47, id attribute)	The venue for the gig.
FK	ArtistId	int	No	FK referenced from entity Artist (table 38, id attribute)	The artist related to the gig.
FK	Country	nvarchar(50)	No	Country and city make a composite FK referenced from entity Location (table 48, country attribute)	The country location of the gig.
FK	City	nvarchar(50)	No	Country and city make a composite FK referenced from entity Location (table	The city location of the gig.

				48, city attribute)	

Data from the Fee entity is used by Gig Management module functions (section 2.2.5., tables 14,15) as well as in the Report management module (section 2.2.9., table 34).

Table 41. FEE entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	FeeId	int	No	IDENTITY (1, 1)	Automatic id for fee entity
	Type	nvarchar(50)	No		Type of the fee.
	AmountWithoutTaxes	int	No		Amount without taxes.
	TaxesPaid	char	No		Contains Y or N value to show whether the taxes for the gig has been paid.
FK	GigId	int	No	FK referenced from entity Gig (table 40, id attribute)	Gig for which the fee was paid.
FK	ArtistId	int	No	FK referenced from entity Artist (table 38, id attribute)	The artist to whom the fee has been paid.

Data from the Attendee entity is used by Gig Management module functions (section 2.2.5., tables 14,16) as well as in the Report management module (section 2.2.9., table 34).

Table 42. ATTENDEE entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	AttendeeId	int	No	IDENTITY (1, 1)	Automatic id for Attendee entity

	Description	nvarchar(255)	Yes		Description of the audience of the gig.
	NumberOfPersons	int	No		Number of persons present in the gig (not counting Guests and Media Representatives)
	AverageAgeRange	nvarchar(30)	Yes		Short description of the age range of the audience, providing a range like “25-40” or description like “mostly adolescents”.
FK	GigId	int	No	FK referenced from entity Gig (table 40, id attribute)	Gig where these attendees are registered to.

Data from the Guests entity is used by Gig Management module functions (section 2.2.5., tables 14,16) as well as in the Report management module (section 2.2.9., table 34).

Table 43. GUESTS entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	GuestsId	int	No	IDENTITY (1, 1)	Automatic id for Guests entity
	NumberOfPersons	int	No		Number of persons present in the “guest list” and checked in the gig.
FK	GigId	int	No	FK referenced from entity Gig (table 40, id attribute)	Gig where these guests where present.

Data from the MediaRepresentative entity is used by the Gig management module functions (section 2.2.5., tables 14,16).

Table 44. MEDIA REPRESENTATIVE entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	MediaRepId	int	No	IDENTITY (1, 1)	Automatic id for Media Representative entity
	Name	nvarchar(100)	No		Name of the media representative
	MediaName	nvarchar(100)	Yes		Name of the media, the person represents.
	MediaType	nvarchar(30)	No		Type of media - radio, magazine, web etc.

MediaRepInGig entity is used to solve the problem of future denormalization of database structure. MediaRepInGig bridges two entities (MediaRepresentative and Gig) between each other.

Table 45. MEDIA REP IN GIG entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	MediaRepInGigId	int	No	IDENTITY (1, 1)	Automatic id for Media Representative which has been present in Gig entity
FK	GigId	int	No	FK referenced from entity Gig (table 40, id attribute)	Id of the gig the media representative has attended.
FK	MediaRepId	int	No	FK referenced from entity MediaRepresentative (table 43, id attribute)	Id of the media representative from the MediaRepresentative entity which has been present in the Gig.

Data from the Tickets entity is used by Gig Management module functions (section 2.2.5., tables 14, 15) as well as in the Report management module (section 2.2.9., table 34).

Table 46. TICKETS entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	TicketsId	int	No	IDENTITY (1, 1)	Automatic id for Tickets entity
	Price	float	No		Average price of one ticket
	Quantity	int	No		Quantity of the tickets sold
	GigId	int	No	FK referenced from entity Gig (table 40, id attribute)	Id of the gig for which tickets are sold.

3.1.4. Venues and corresponding entities

This subsection contains detailed information about table design of Venue entity (table 47) and related entities – Location (table 48) and Backline (table 49).

Data from the Venue entity is used by the Venue management module functions (section 2.2.8., tables 29-31), Gig management module functions (section 2.2.5., tables 14, 16) as well as in Report management module (section 2.2.9., table 34).

Table 47. VENUE entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	VenueId	int	No	IDENTITY (1, 1)	Automatic id for Venue entity
	Name	nvarchar(100)	No		Name of the venue
	Capacity	int	Yes		Max capacity of the venue for standard seated/standing

	Type	nvarchar(30)	No		Type of the venue
	Notes	nvarchar(255)	Yes		Body of the notes regarding the venue
FK	Country	nvarchar(50)	No	Country and city make a composite FK referenced from entity Location (table 48, country attribute)	The country location of the venue.
FK	City	nvarchar(50)	No	Country and city make a composite FK referenced from entity Location (table 48, city attribute)	The city location of the venue.

Data from the Location entity is used by the Venue management module (section 2.2.8, tables 29-31), Gig management module functions (section 2.2.5., tables 14, 15), as well as the Report management module (section 2.2.9., table 34).

Table 48. LOCATION entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	City	nvarchar(50)	No		City where the gig is taking place
PK	Country	nvarchar(50)	No		Country in which is the city where the gig is taking place
	Region	nvarchar(50)	Yes		Region in which is the country where the gig is taking place

Data from the Backline entity is used by Venue management module (section 2.2.8., tables 29-31), Gig Management module functions (section 2.2.5., tables 14,15) as well as in Report management module (section 2.2.9., table 34).

Table 49. BACKLINE entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	BacklineId	int	No	IDENTITY (1, 1)	Automatic id for Backline entity
	Description	nvarchar(255)	No		Description of the backline in the venue
	Costs	int	Yes		Costs of the backline in the venue, if available and if costs for artist
FK	VenueId	int	No	FK referenced from entity Venue (table 48, id attribute)	Venue in which the backline is situated

3.1.5. Merchandise entity

This subsection contains detailed information about table design of Merchandise entity (table 50).

Data from the Merchandise entity is used by Merchandise management module functions (section 2.2.8., tables 27-28), Gig management module functions (section 2.2.5., tables 14, 15, 16) as well as in Report management module (section 2.2.9., table 34).

Table 50. MERCHANDISE entity – description

Key	Column name	Data type	Nullable	Other parameters	Description
PK	MerchandiseId	int	No	IDENTITY (1, 1)	Automatic id for Merchandise entity
	Description	nvarchar(255)	No		Description of the merchandise item
	Price	int	Yes		Price of the merchandise item
	Sold	char	Yes		Contains Y or N value to show whether the

					mentioned merchandise item has been sold.
FK	GigId	int	Yes	FK referenced from entity Gig (table 40, id attribute)	Gig in which the value is sold

3.1.6. Announcements and corresponding entities

This subsection contains detailed information about table design of Announcement entity (table 51) as well as related entities - Language (table 52) and Type (table 53).

Data from the Announcement entity is used by Announcement management module functions (section 2.2.6., tables 21-26).

Table 51. ANNOUNCEMENT entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	AnnouncementId	int	No	IDENTITY (1, 1)	Automatic id for Announcement entity
	Created	date	No		Date when the announcement is created
	Header	nvarchar(100)	No		Header of the announcement
	Message	nvarchar)(255)	No		Message body of the announcement
FK	UserId	int	No	FK referenced from entity User (table 36, id attribute)	User which has created the announcement
FK	LanguageId	int	No	FK referenced from entity Language (table 52, id attribute)	Language in which the announcement is created

FK	TypeId	int	No	FK referenced from entity Gig (table 53, id attribute)	Type of the announcement

Data from the Language entity is used by the Announcement management module functions (section 2.2.6., tables 21-26).

Table 52. LANGUAGE entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	LanguageId	int	No	IDENTITY (1, 1)	Automatic id for Language entity
	Name	nvarchar(50)	No		Language name
	LanguageCod e	nvarchar(50)	No		Language code according to ISO 639-1:2002 standard

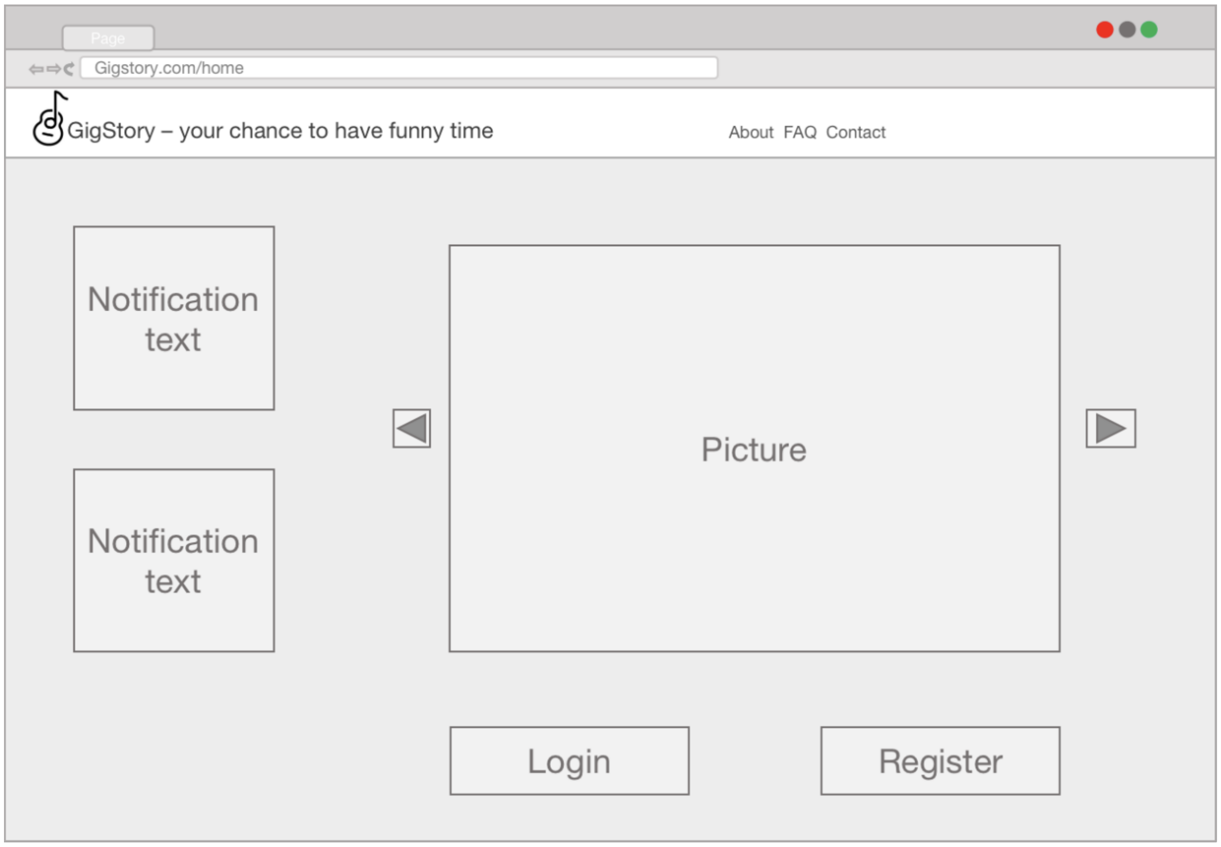
Data from the Type entity is used by the Announcement management module functions (section 2.2.6., tables 21-26).

Table 52. TYPE entity – description

Key	**Column name**	**Data type**	**Nullable**	**Other parameters**	**Description**
PK	TypeId	int	No	IDENTITY (1, 1)	Automatic id for Type entity
	Name	nvarchar(50)	No		Name of the announcement type
	Description	nvarchar(255)	No		Description of the announcement type

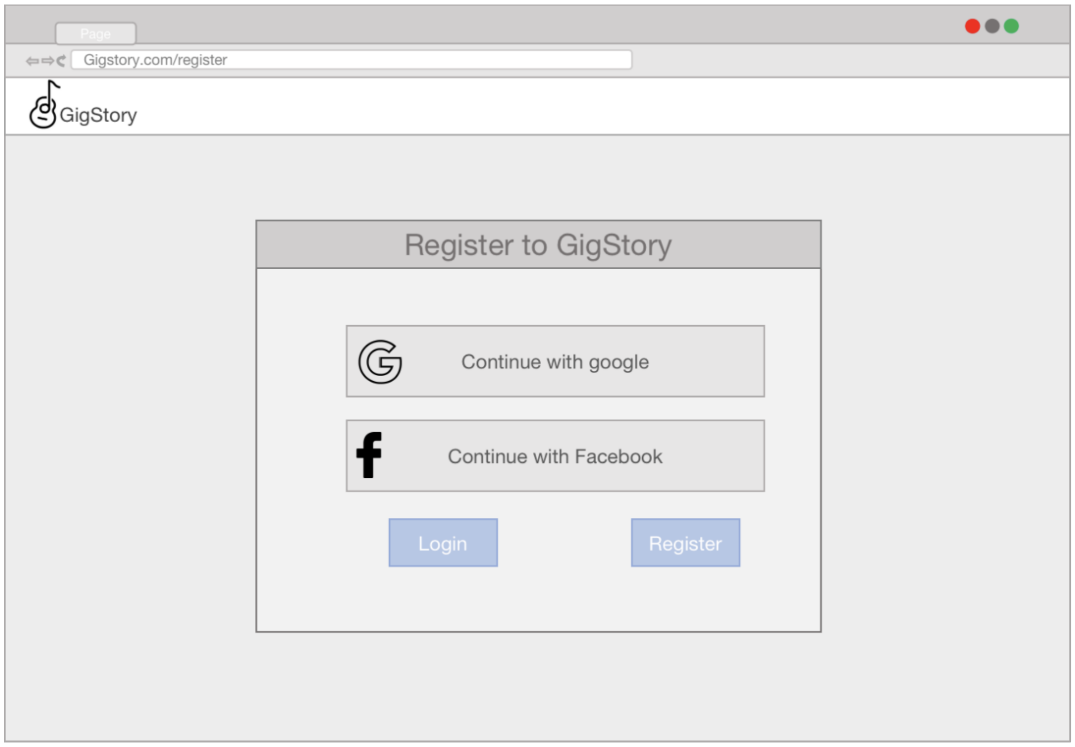
3.2. Partial Design of User Interfaces

The main page (homepage) (pic. 14) includes public announcements, generated by function in the Announcements module (section 2.2.6., table 21). From this page it's possible to either register or login into the system (section 2.2.3., tables 2-3). The very same page is available for User role (according to roles described in section 1.5.) after login to the system.



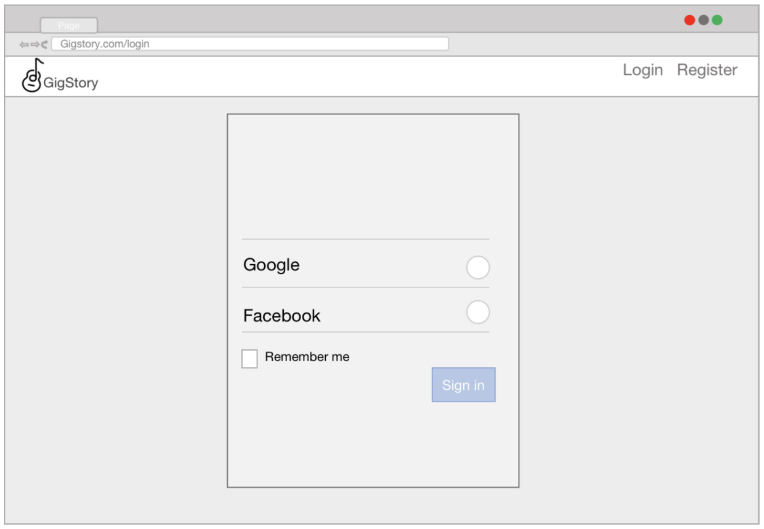
Pic. 14. Main page (Unregistered user)

Registration form (pic. 15) includes fields required by functional requirement in section 2.2.3., table 2.



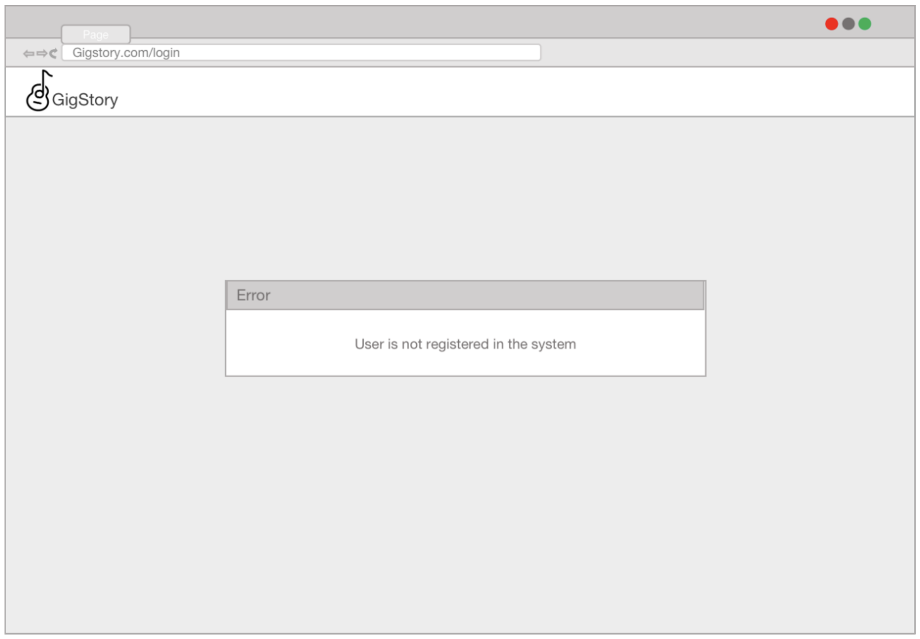
Pic. 15. Registration form

Login form (pic. 16) includes fields required by functional requirement in section 2.2.3., table 3.



Pic. 16. Login form

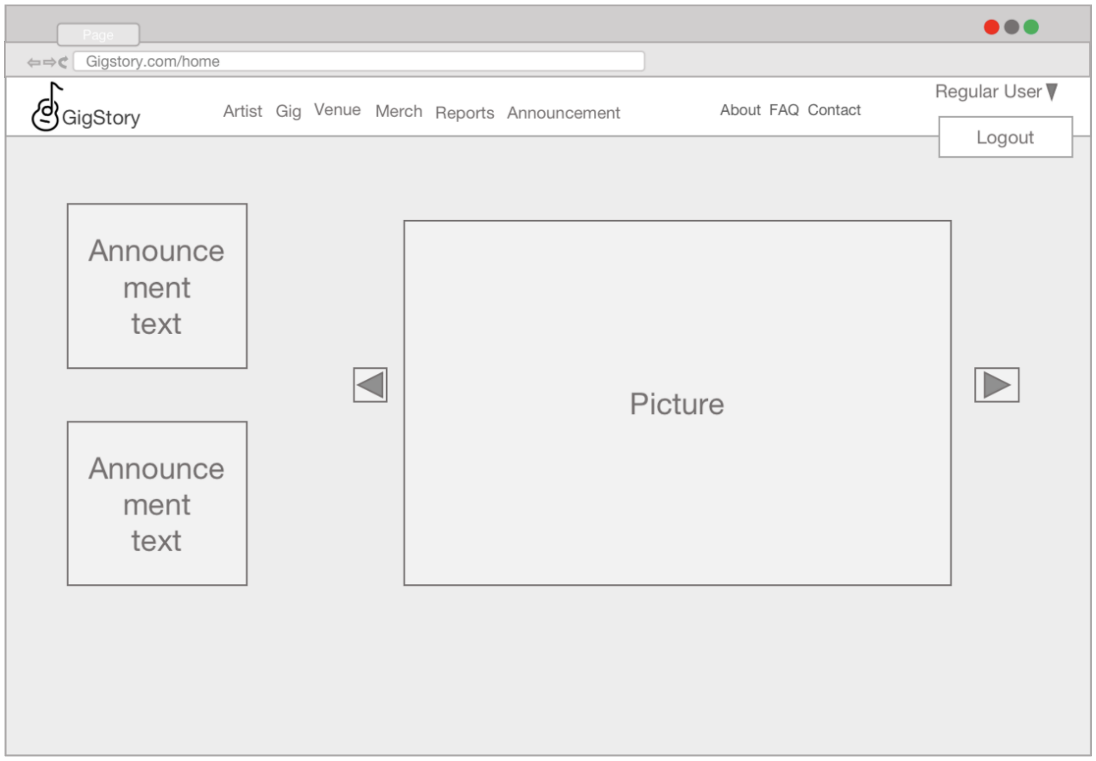
After login if you have an error in email (according to roles described in section 1.5.) will get an error message (pic. 17) based on functional requirements in section 2.2.3., table 3.



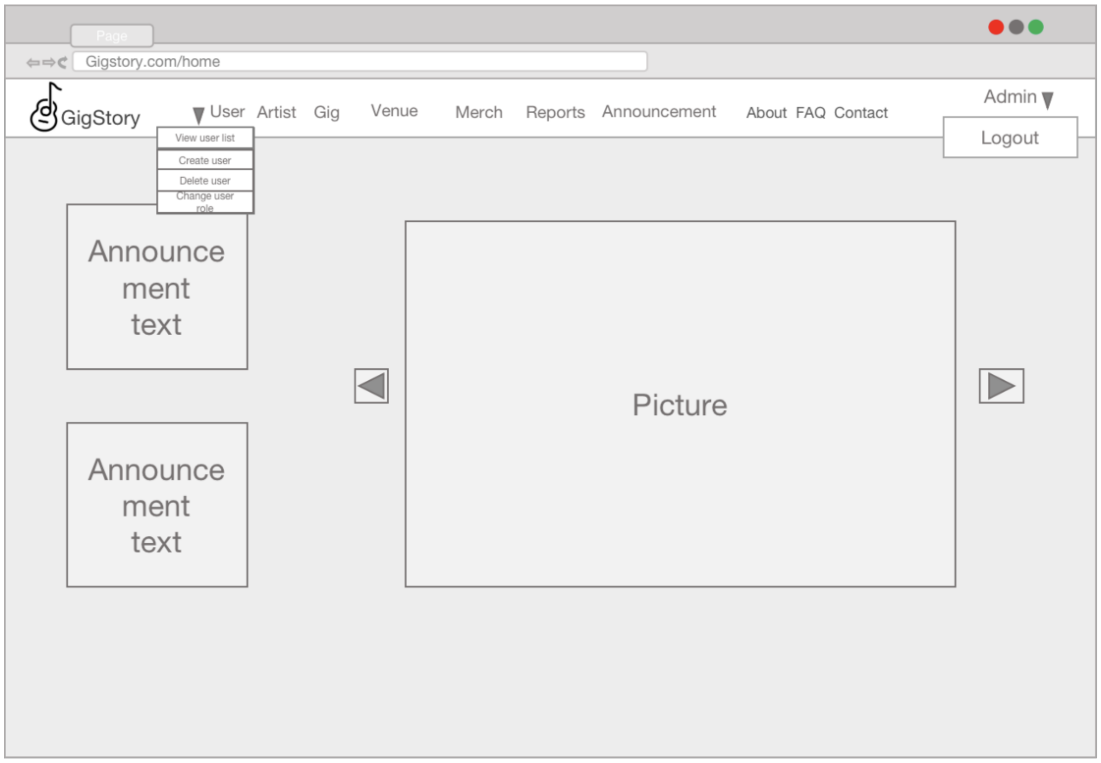
User is not registered in the system

Pic. 17. Error message for login form

Regular user don't have rights to add other users (pic.18). Admin has a list of admin-only actions (pic. 19). They both showed as main pages.

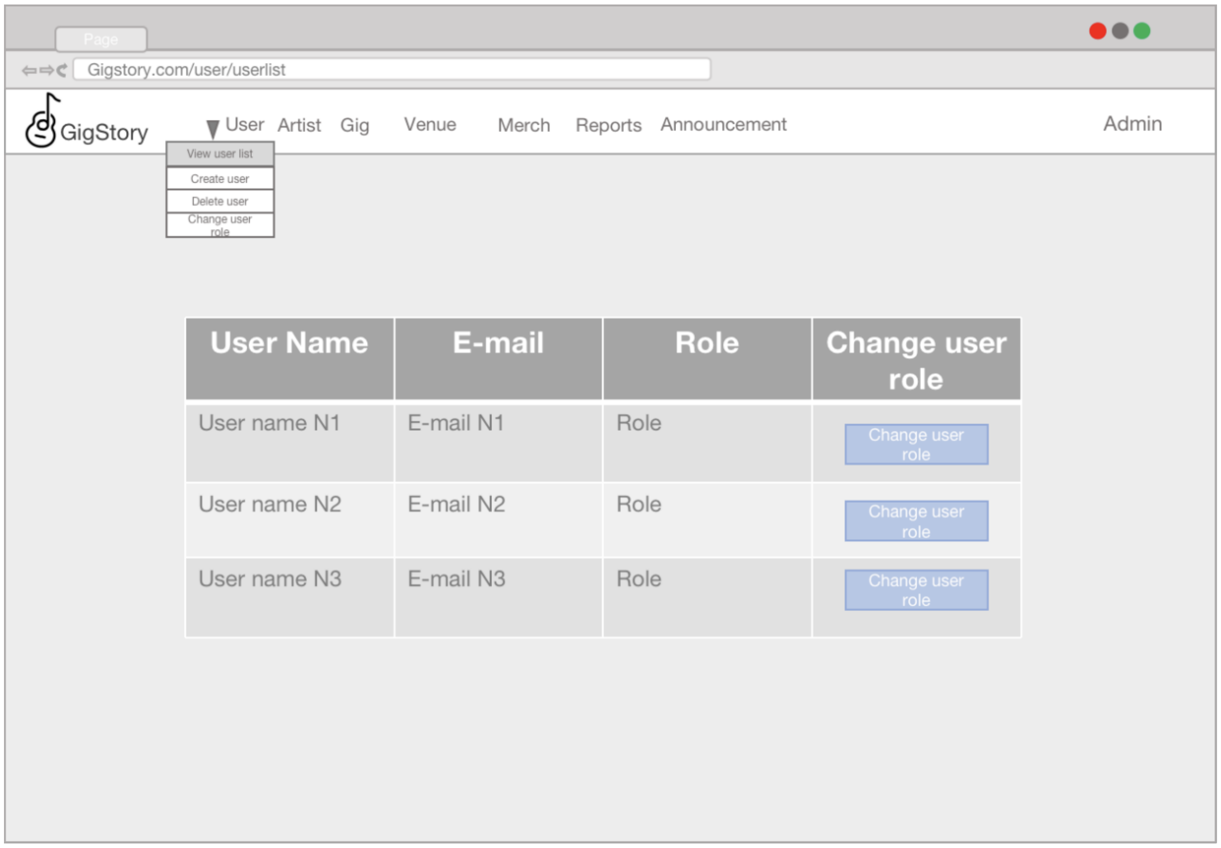


Pic. 18. Main page (Regular user)



Pic. 19. Main page (Admin)

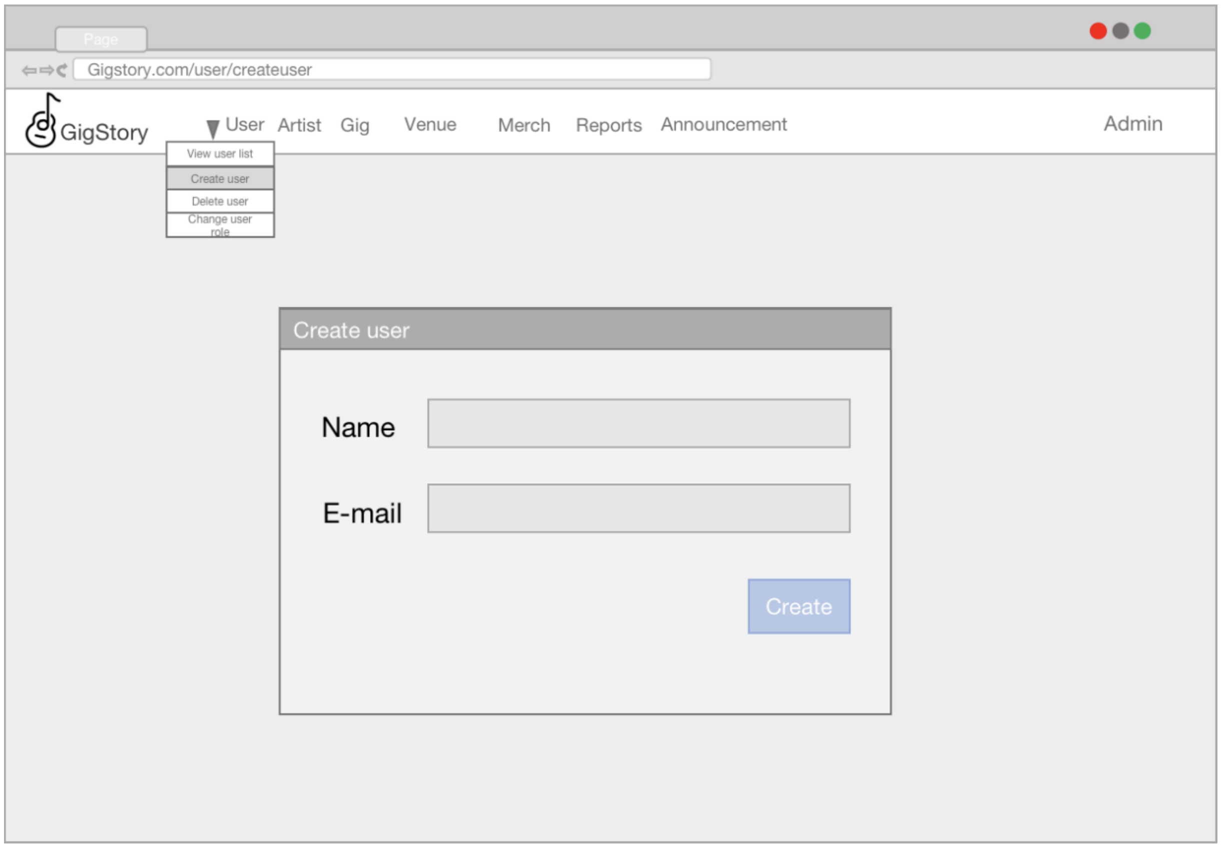
On the pic. 20 see the list of all users created by the admin with corresponding user role. Based on functional requirements in section 2.2.3, table 6.



User Name	E-mail	Role	Change user role
User name N1	E-mail N1	Role	Change user role
User name N2	E-mail N2	Role	Change user role
User name N3	E-mail N3	Role	Change user role

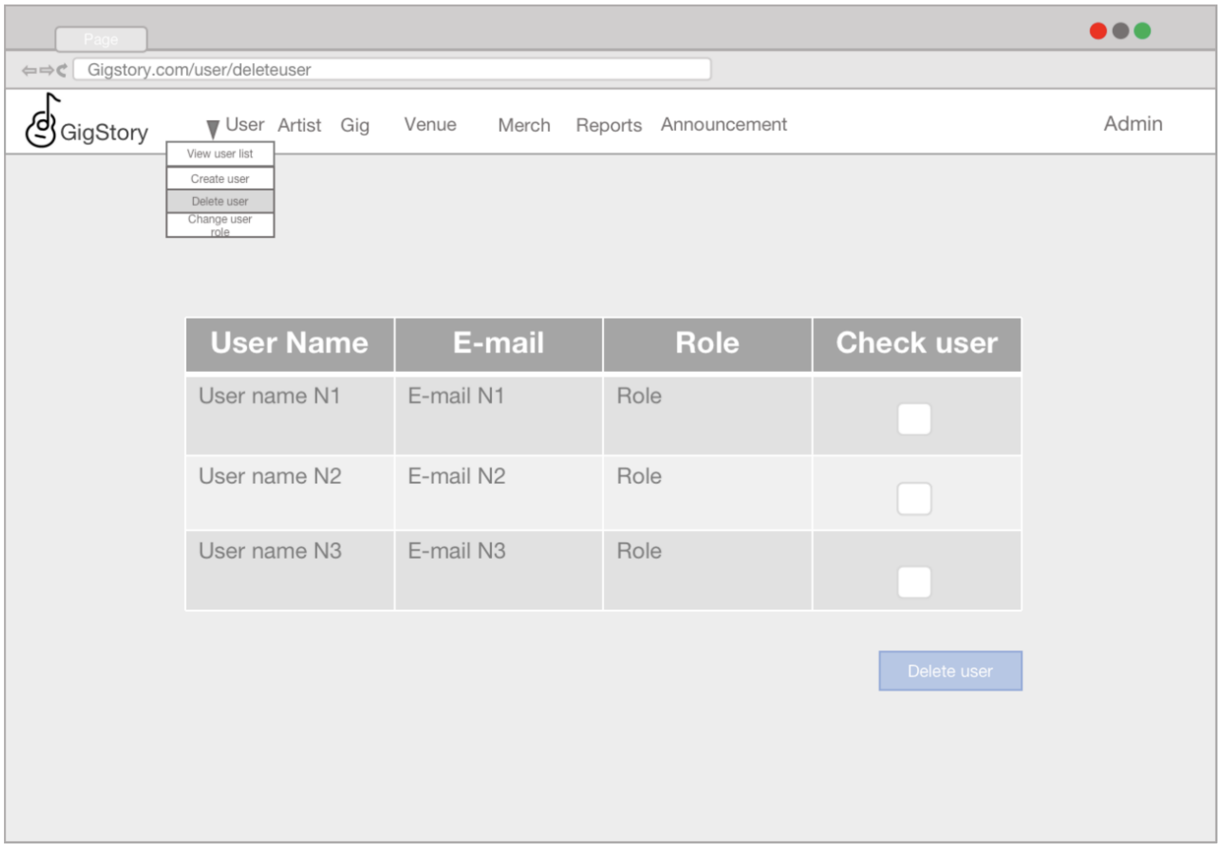
Pic. 20. View user list (admin)

On the pic. 21 system should allow admin to create a new user. Based on functional requirements in section 2.2.3, table 4.



Pic. 21. Create new user (admin)

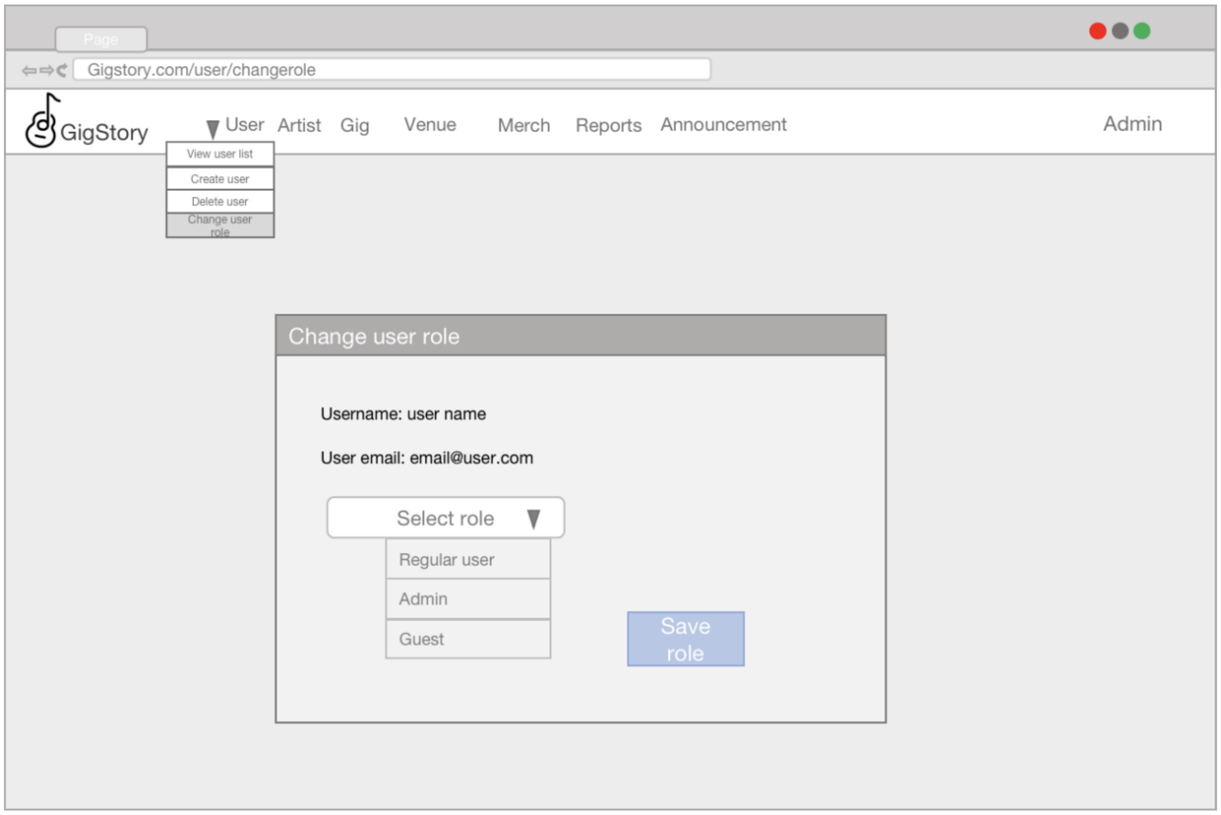
On the pic. 22 deletes a user, having the role “Regular user” or “Guest”, created by admin. Based on functional requirements in section 2.2.3, table 8.



User Name	E-mail	Role	Check user
User name N1	E-mail N1	Role	☐
User name N2	E-mail N2	Role	☐
User name N3	E-mail N3	Role	☐

Pic. 22. Delete user (admin)

On the pic.23 changes the role of the selected user. Based on functional requirements in section 2.2.3, table 7.



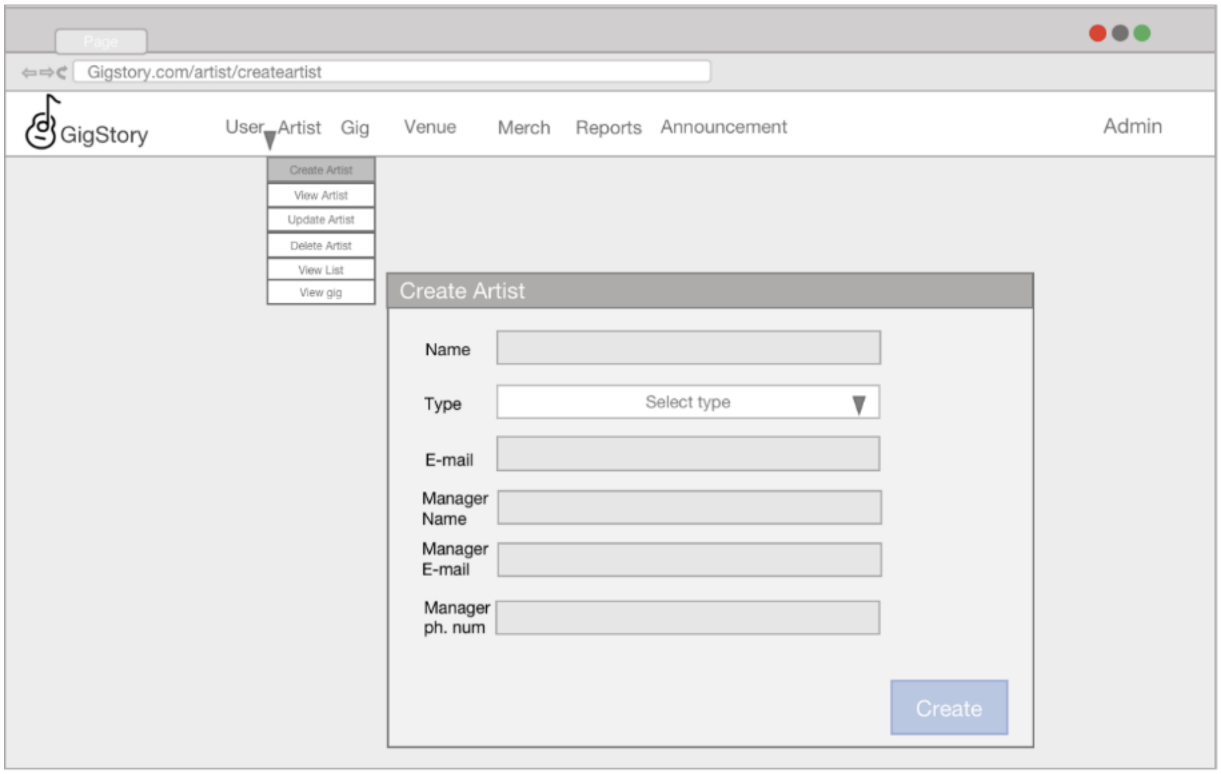
Username: user name

User email: email@user.com

Pic. 23. Change user role (admin)

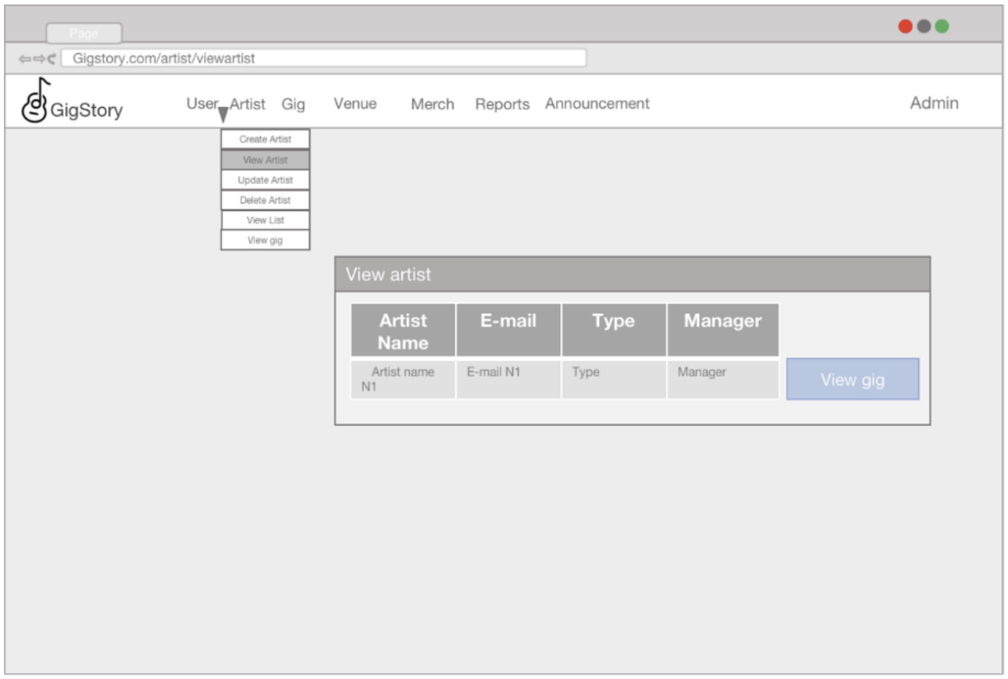
The regular and admin users have the same possibilities to create artist, view artist, update artist, view artist list and view artist gig list. Only admin have possibilities to delete artist.

On pic. 24 system should allow regular and admin users to create a new artist. Based on functional requirements in section 2.2.4, table 9.



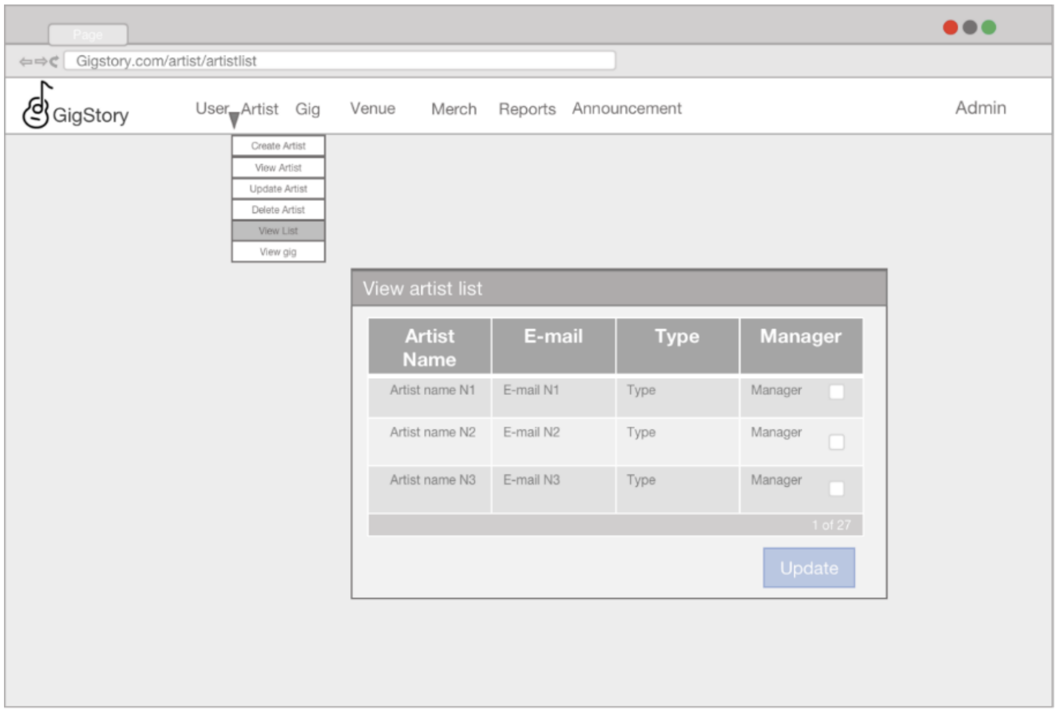
Pic. 24. Create artist (admin)

On pic. 25 and 25.1 the system should allow regular users and admin users to view artist and list of all artists . Based on functional requirements in section 2.2.4, table 11.



Artist Name	E-mail	Type	Manager
Artist name N1	E-mail N1	Type	Manager

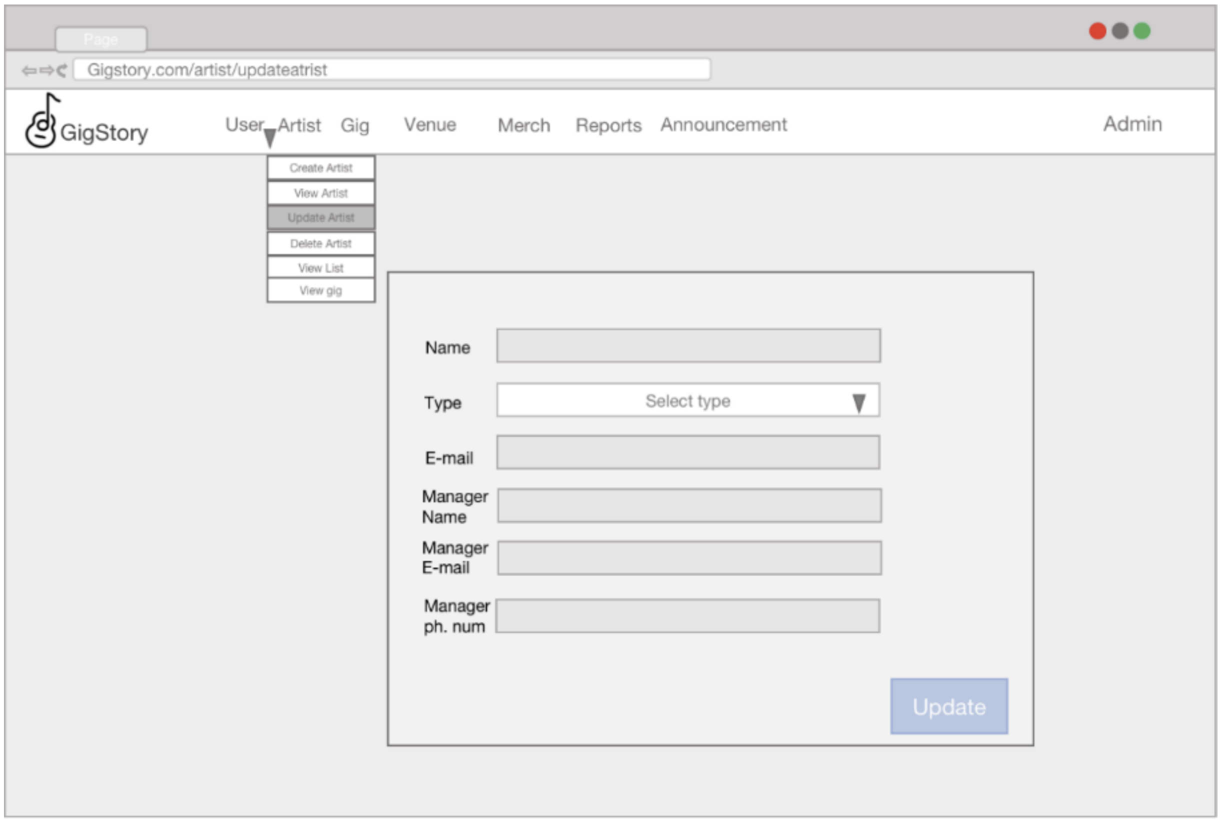
Pic. 25. View artist (admin)



Artist Name	E-mail	Type	Manager
Artist name N1	E-mail N1	Type	Manager
Artist name N2	E-mail N2	Type	Manager
Artist name N3	E-mail N3	Type	Manager

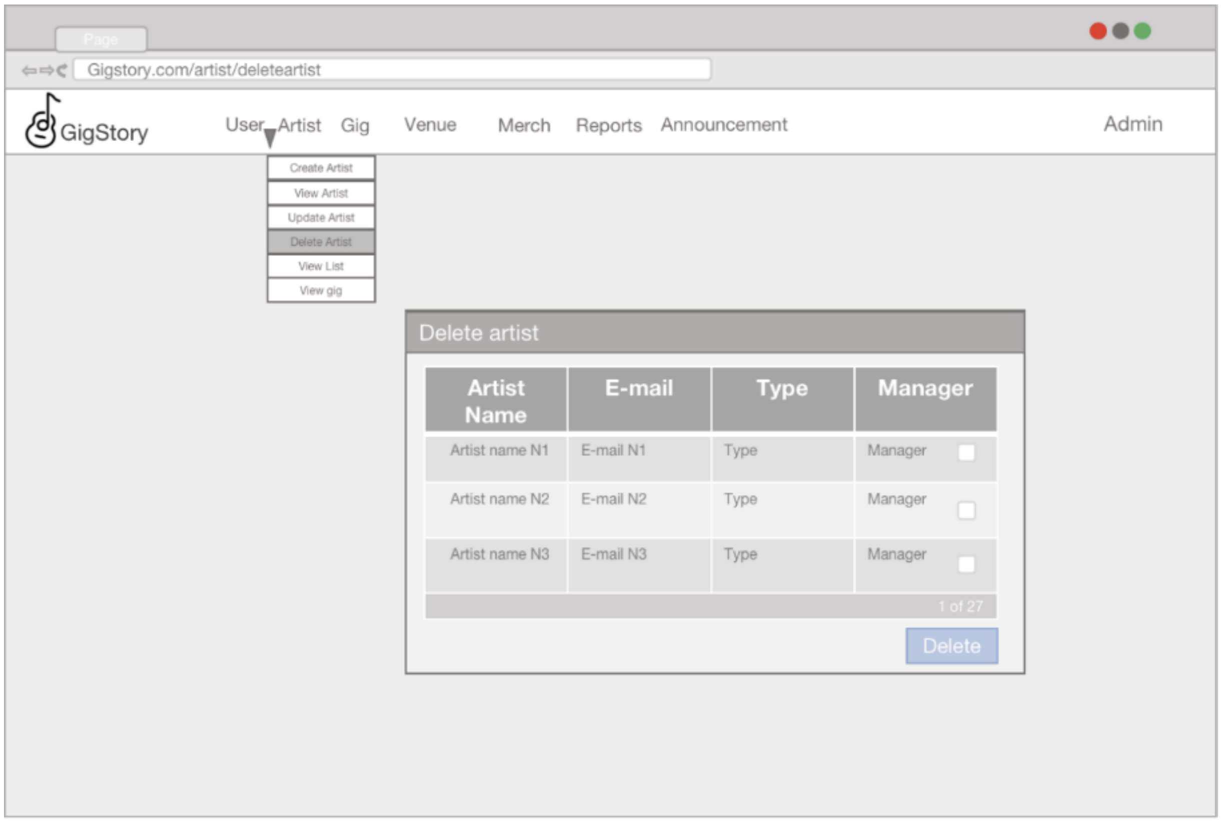
Pic. 25.1 View artist list (admin)

On pic. 26 system should allow regular and admin users to update information in the artist profile. Based on functional requirements in section 2.2.4, table 10.



Pic. 26. Update artist (admin)

On pic. 27 system should allow admin users to delete an artist profile. Based on functional requirements in section 2.2.4, table 12.

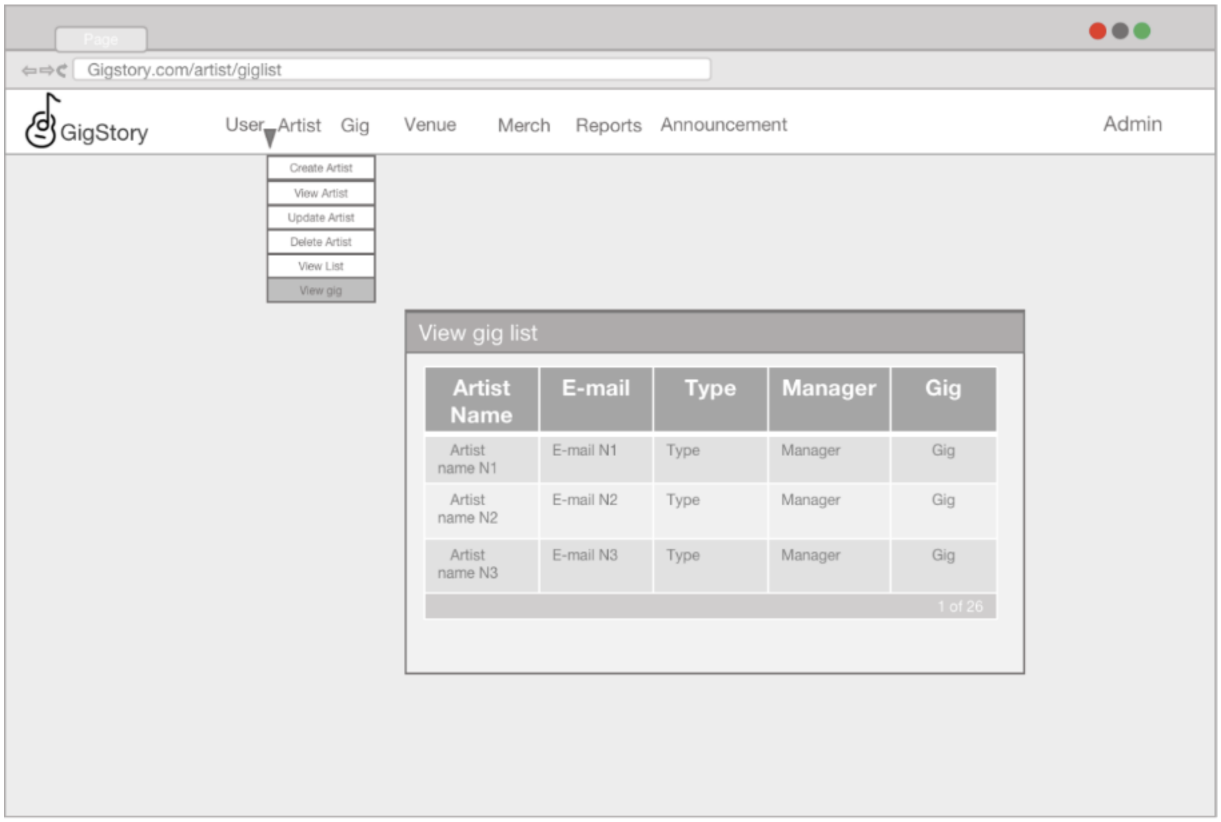


Artist Name	E-mail	Type	Manager
Artist name N1	E-mail N1	Type	Manager
Artist name N2	E-mail N2	Type	Manager
Artist name N3	E-mail N3	Type	Manager

1 of 27

Pic. 27. Delete artist (admin)

On pic. 28 should allow regular and admin users to view a list of gigs for an artist. Based on functional requirements in section 2.2.4, table 13.

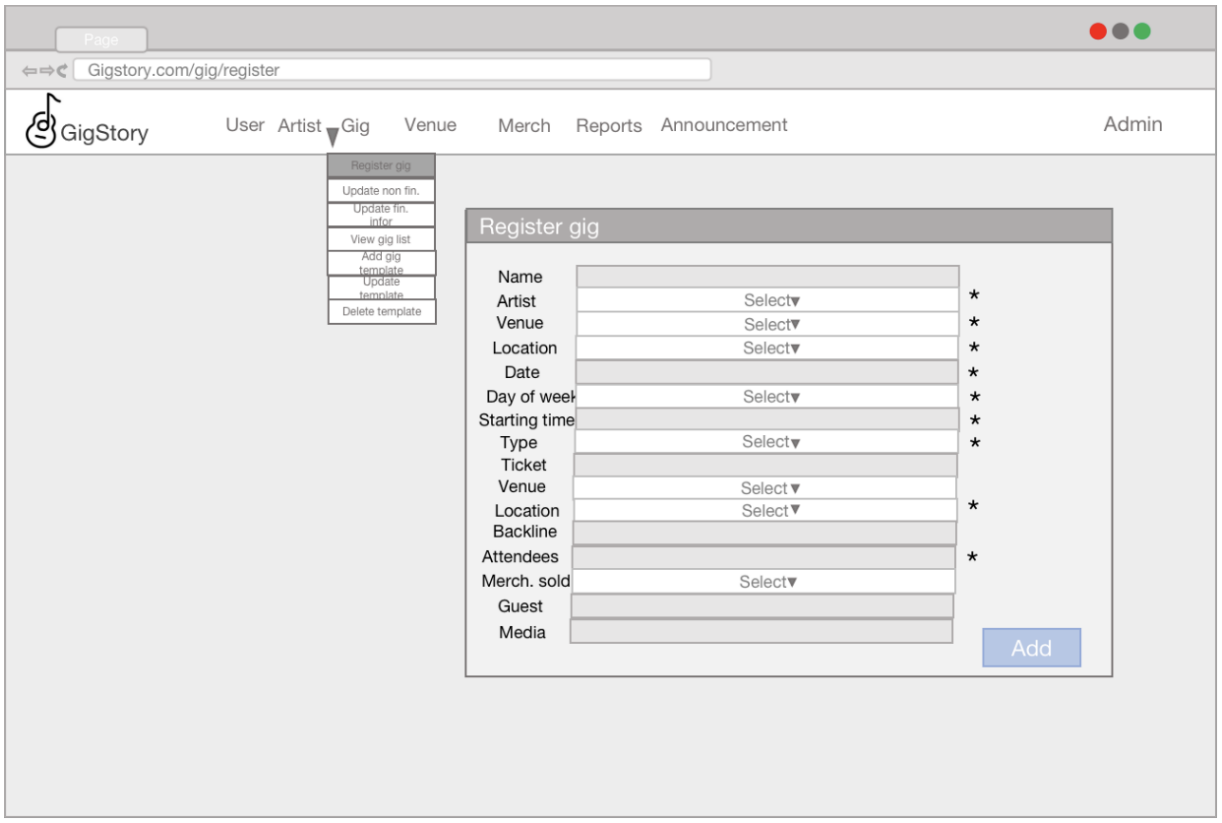


Artist Name	E-mail	Type	Manager	Gig
Artist name N1	E-mail N1	Type	Manager	Gig
Artist name N2	E-mail N2	Type	Manager	Gig
Artist name N3	E-mail N3	Type	Manager	Gig

1 of 26

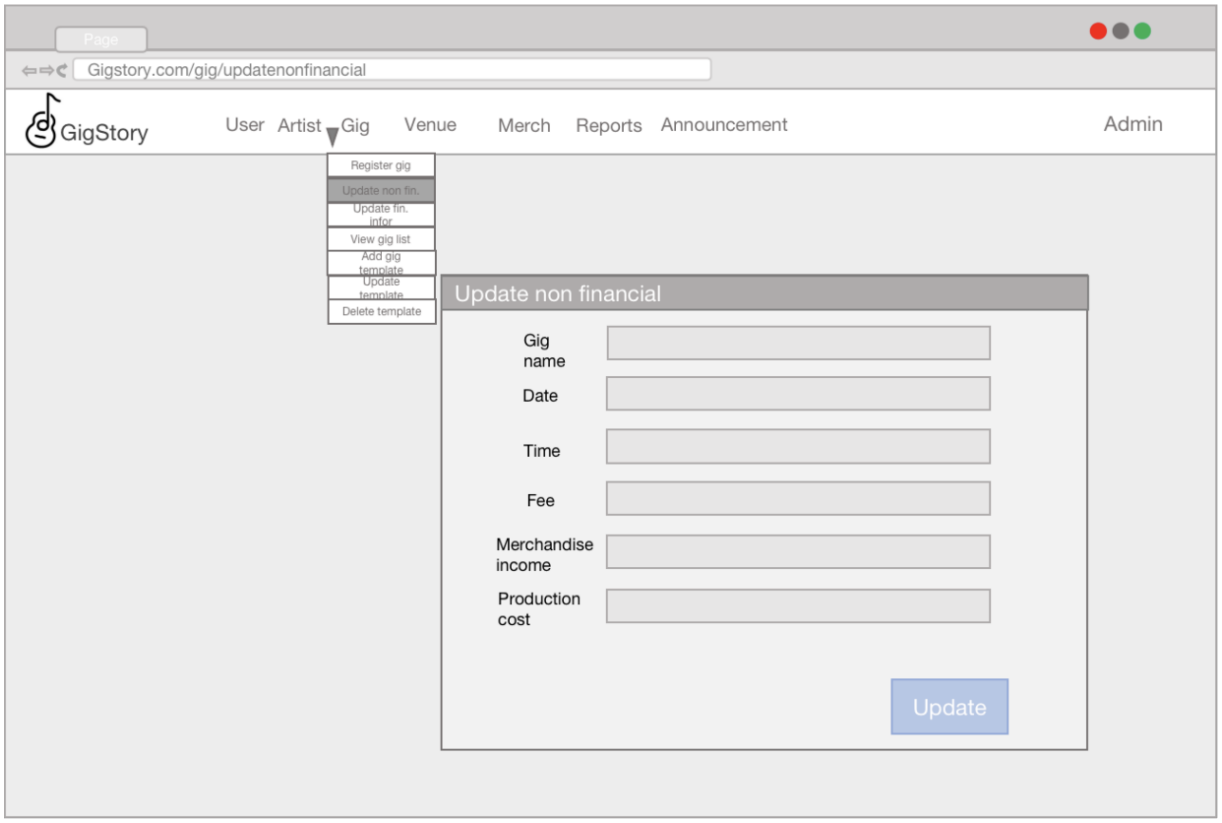
Pic. 28. View gig list (admin)

On pic. 29 system should allow regular and admin users to register a new gig. Based on functional requirements in section 2.2.5, table 14.



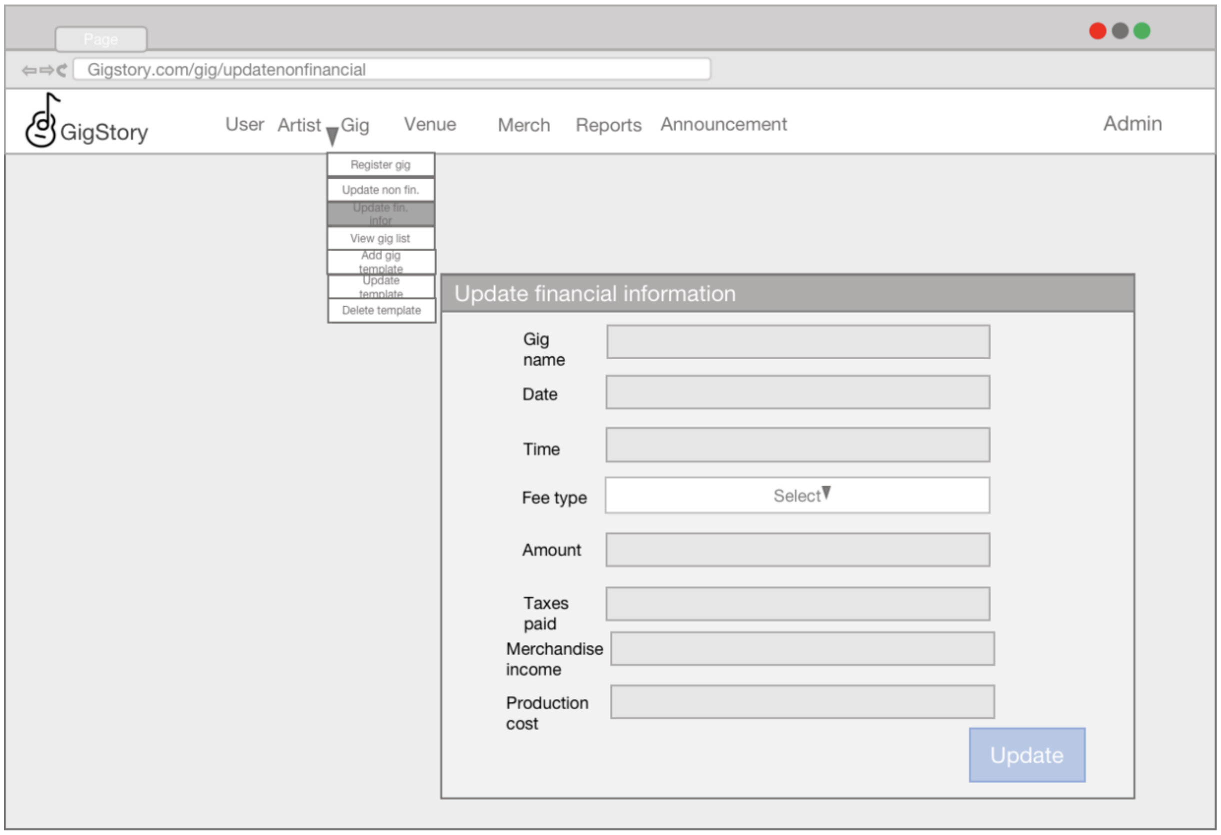
Pic. 29. Register new gig (admin)

On pic. 30 the system should allow regular and admin users to edit any non-financial information of a registered gig. Based on functional requirements in section 2.2.5, table 16.



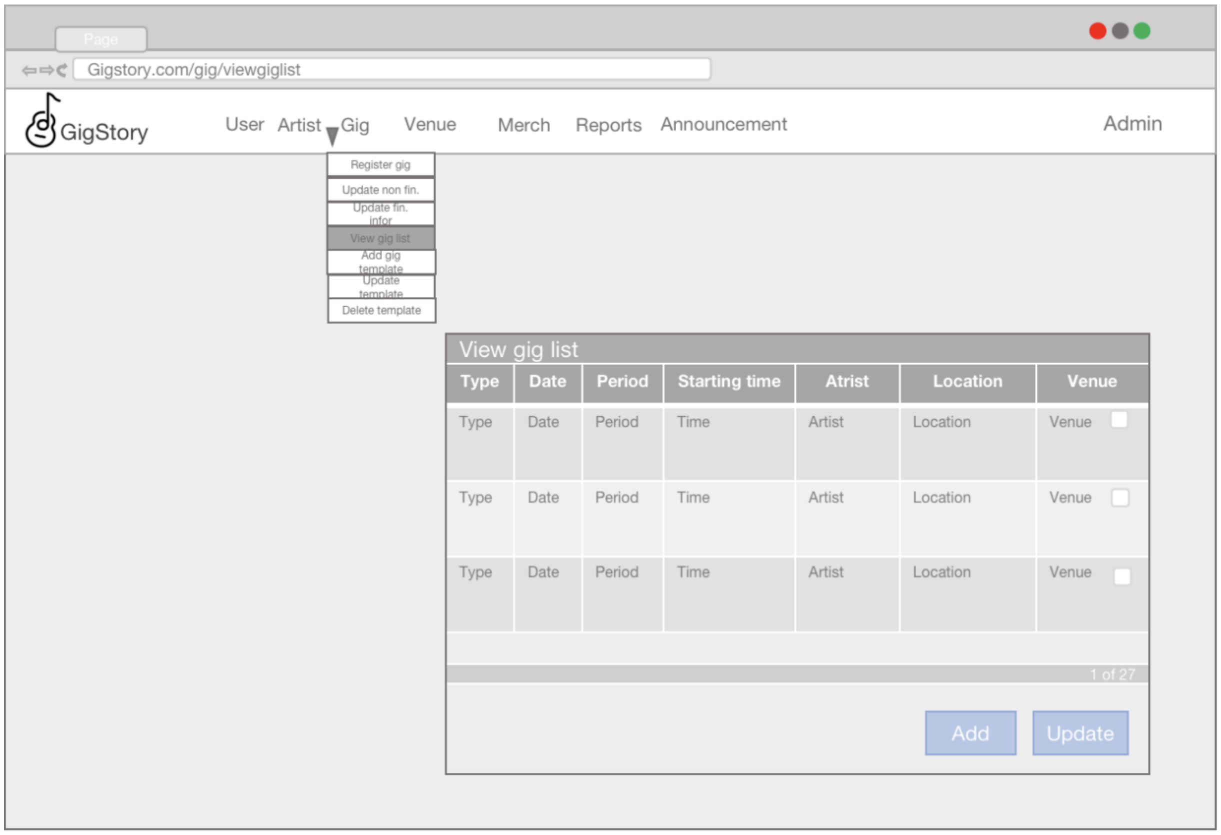
Pic. 30. Update non-financial information of the gig (admin)

On pic. 31 the system should allow admin users to add financial information to a registered gig. Based on functional requirements in section 2.2.5, table 15.



Pic. 31 Update financial information of the gig (admin)

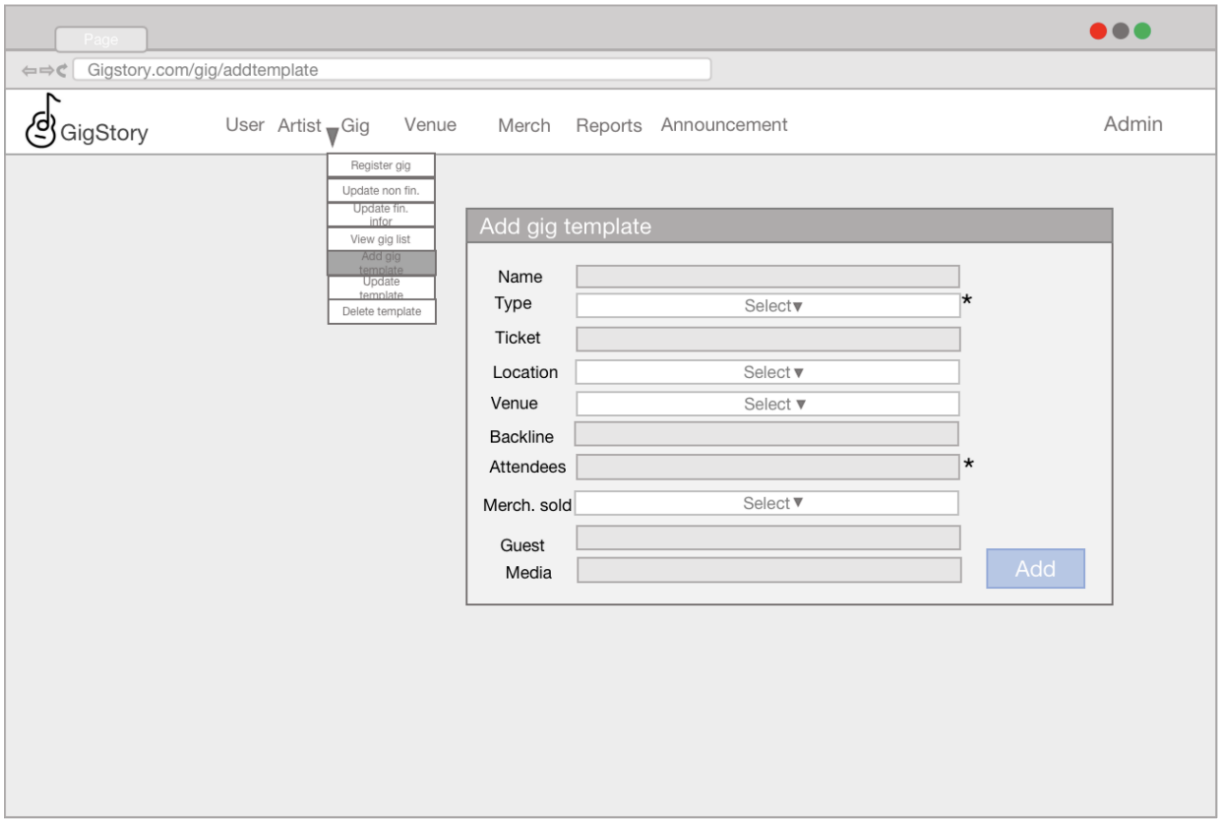
On the pic. 32 system should allow regular and admin users to view a list of registered gigs. Based on functional requirements in section 2.2.5, table 20.



Type	Date	Period	Starting time	Atrist	Location	Venue
Type	Date	Period	Time	Artist	Location	Venue
Type	Date	Period	Time	Artist	Location	Venue
Type	Date	Period	Time	Artist	Location	Venue

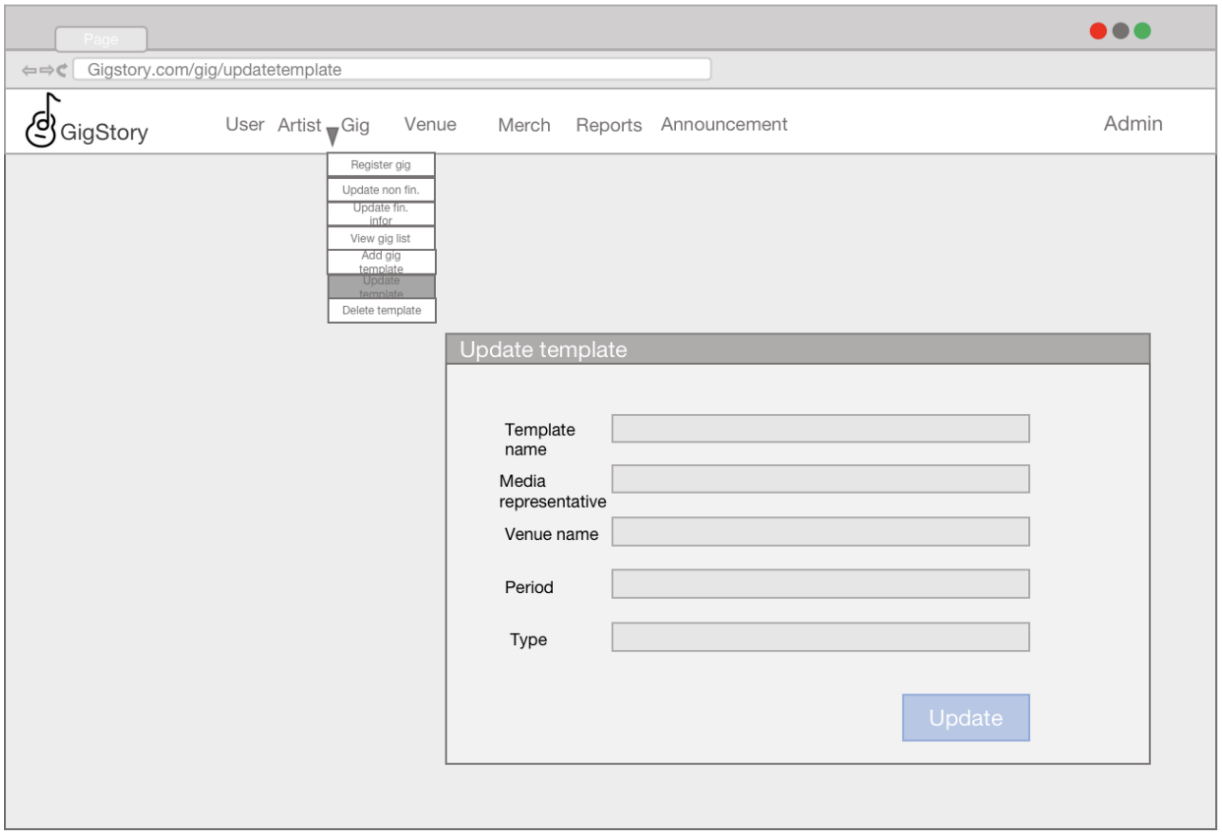
Pic. 32. View gig list (admin)

On pic. 33 system s should allow admin users to add gig records with attribute "isTemplate" = true. Based on functional requirements in section 2.2.5, table 17.



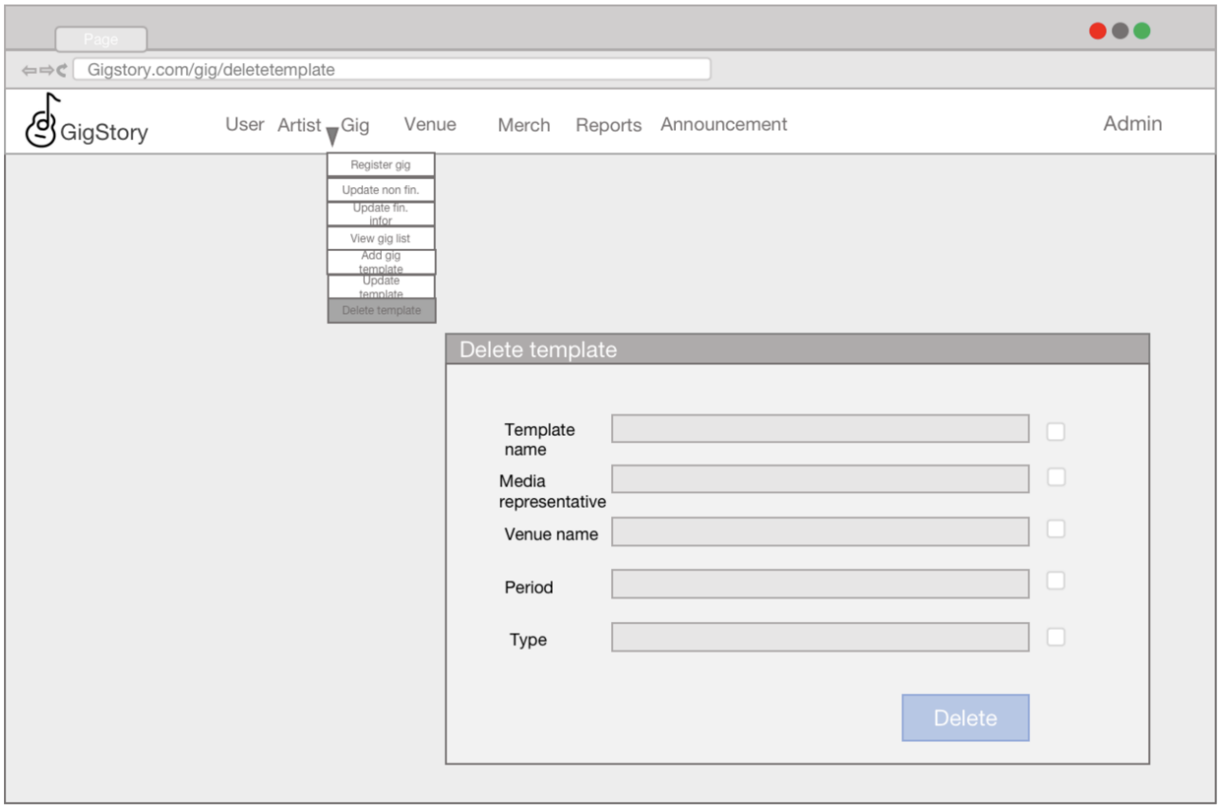
Pic. 33. Add gig template (admin)

On pic. 34 system should allow admin users to edit the gig records saved with “isTemplate” = true. Based on functional requirements in section 2.2.5, table 18.



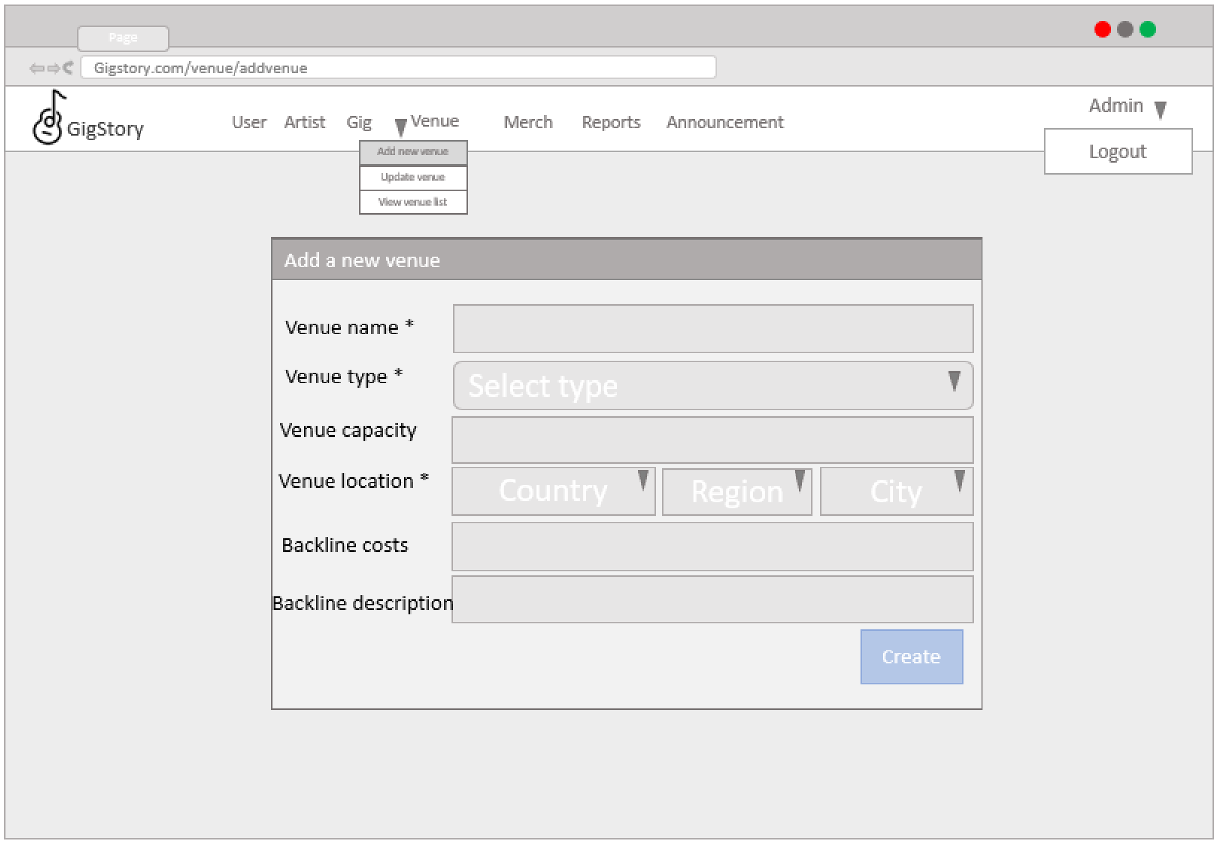
Pic. 34. Update gig template (admin)

On pic. 35 the system should allow admin users to delete the gig records saved with “isTemplate” = true. Based on functional requirements in section 2.2.5, table 19.



Pic. 35. Delete gig template (admin)

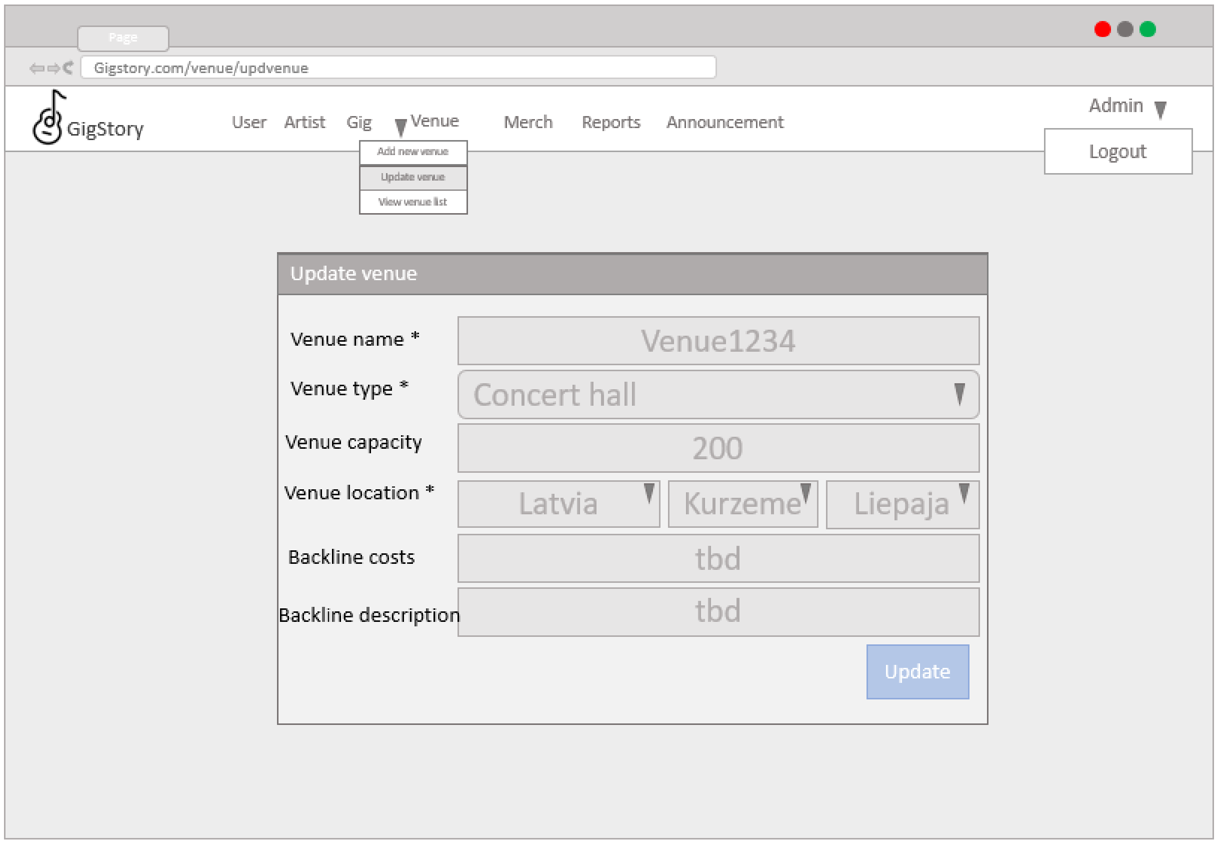
On pic. 36 the system should allow admin users to add a new venue. Mandatory input is marked with the star - Venue name, Venue type, and Venue location. Based on functional requirements in section 2.2.8, table 29.



Venue name *	
Venue type *	Select type
Venue capacity	
Venue location *	Country / Region / City
Backline costs	
Backline description	

Pic. 36. Add new venue (admin)

On pic. 37 the system should allow admin users to update a venue. Mandatory input is the same as for adding the venue, and the admin is allowed to change all the input fields if needed. Based on functional requirements in section 2.2.8, table 30.

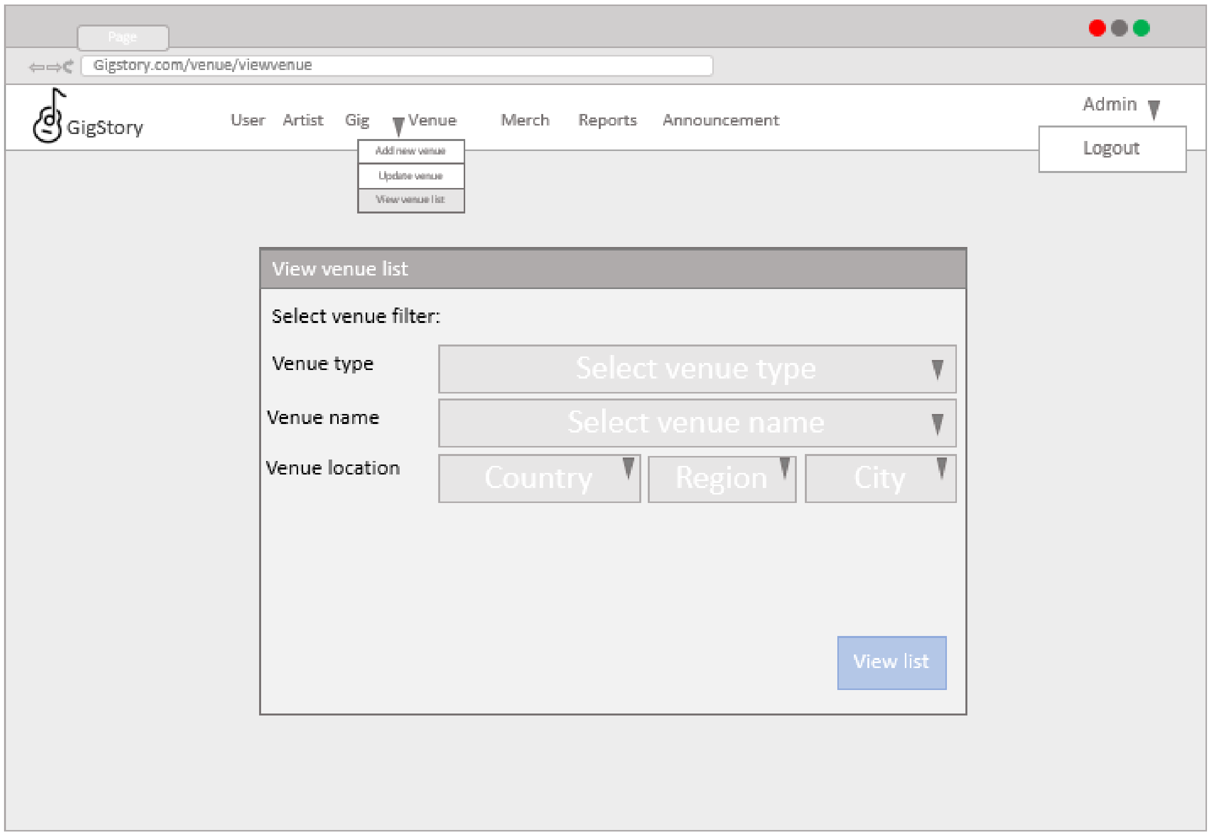


Update venue

Venue name *	Venue1234
Venue type *	Concert hall
Venue capacity	200
Venue location *	Latvia, Kurzeme, Liepaja
Backline costs	tbd
Backline description	tbd

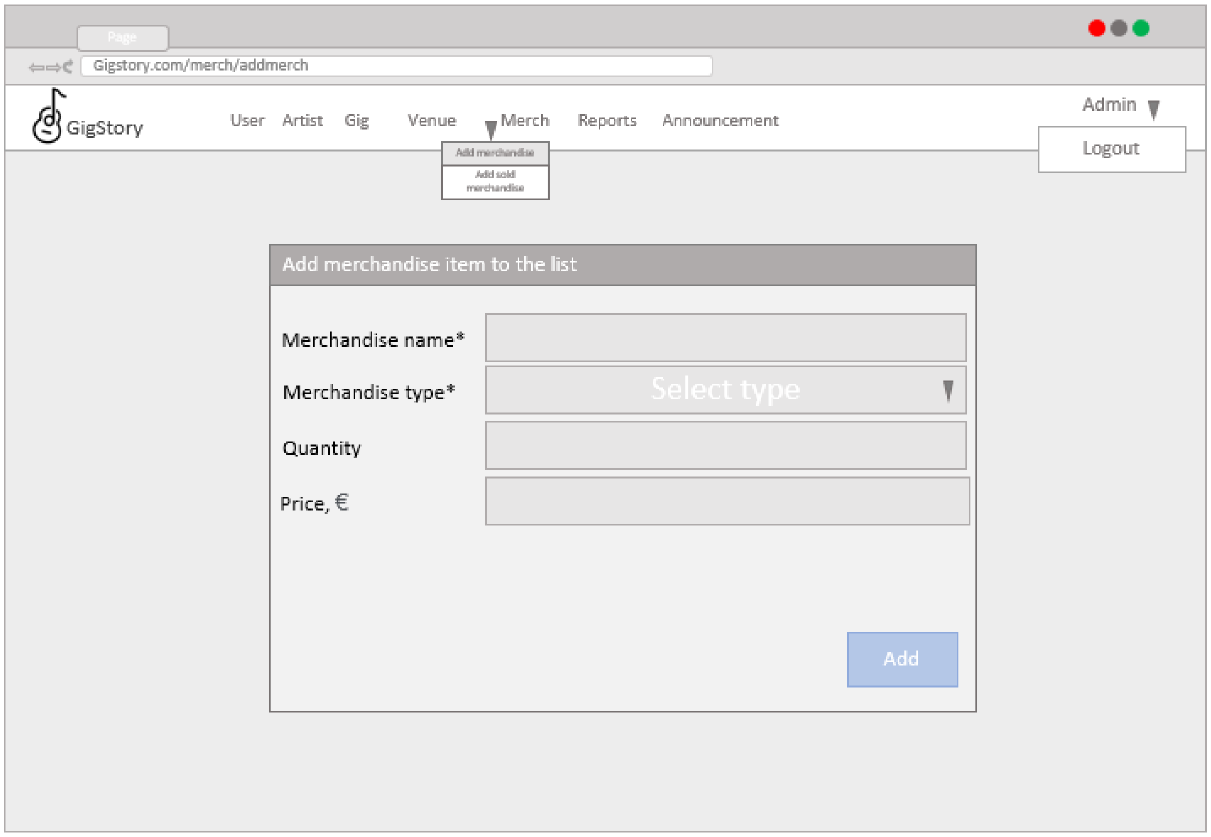
Pic. 37. Update venue (admin)

On pic. 38 the system should allow admin users to view the list of all the venues. The selection of needed venues can be done by either or all of the dropdown fields - Venue Type, Venue Name, and/or Venue Location. Based on functional requirements in section 2.2.8, table 31.



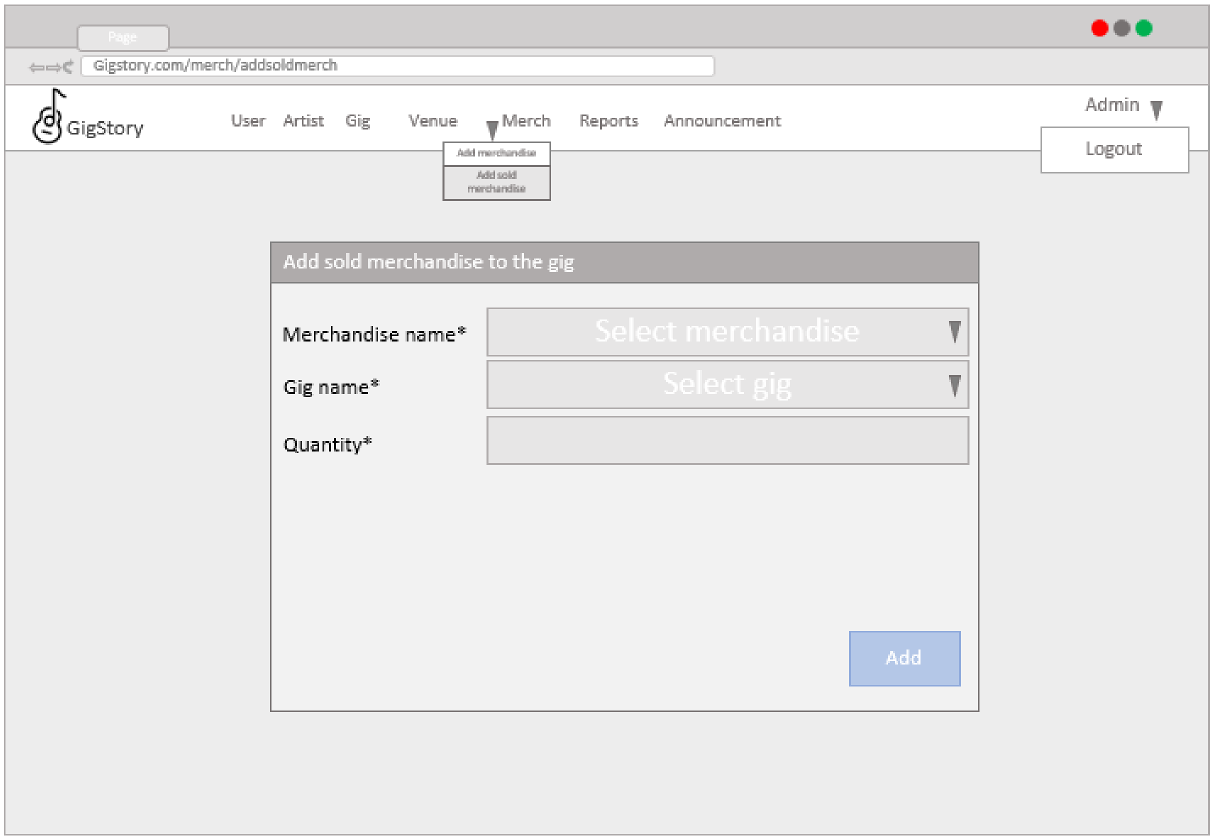
Pic. 38. View venue list (admin)

On pic. 39 the system should allow admin users to add any possible merchandise to the database. Mandatory parts are Merchandise name and type, but the price and quantity in stock can also be added. Based on functional requirements in section 2.2.7, table 27.



Pic. 39. Add merchandise (admin)

On pic. 40 the system should allow admin users to add sold merchandise to a gig. The admin should input the merchandise name, the Gig name and the quantity of sold merchandise. Based on functional requirements in section 2.2.7, table 28.

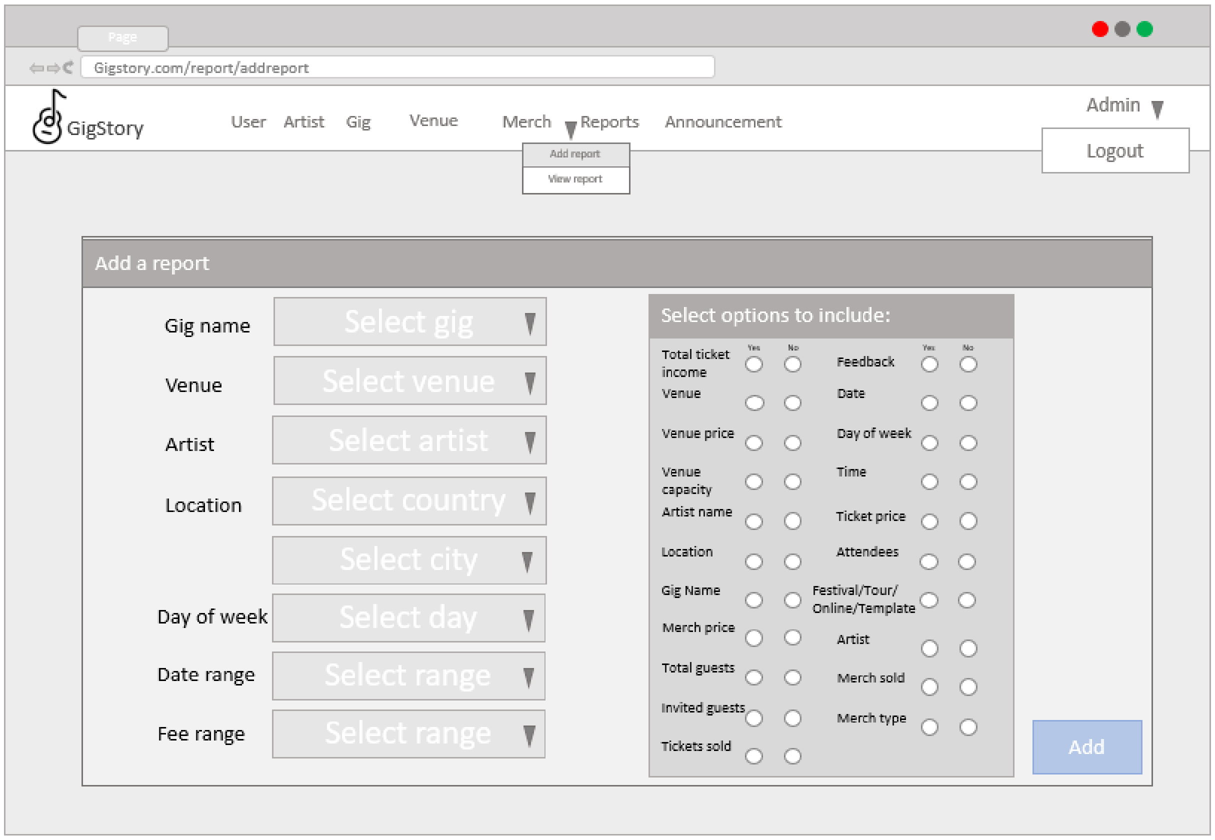


Add sold merchandise to the gig

Merchandise name*	Select merchandise
Gig name*	Select gig
Quantity*	

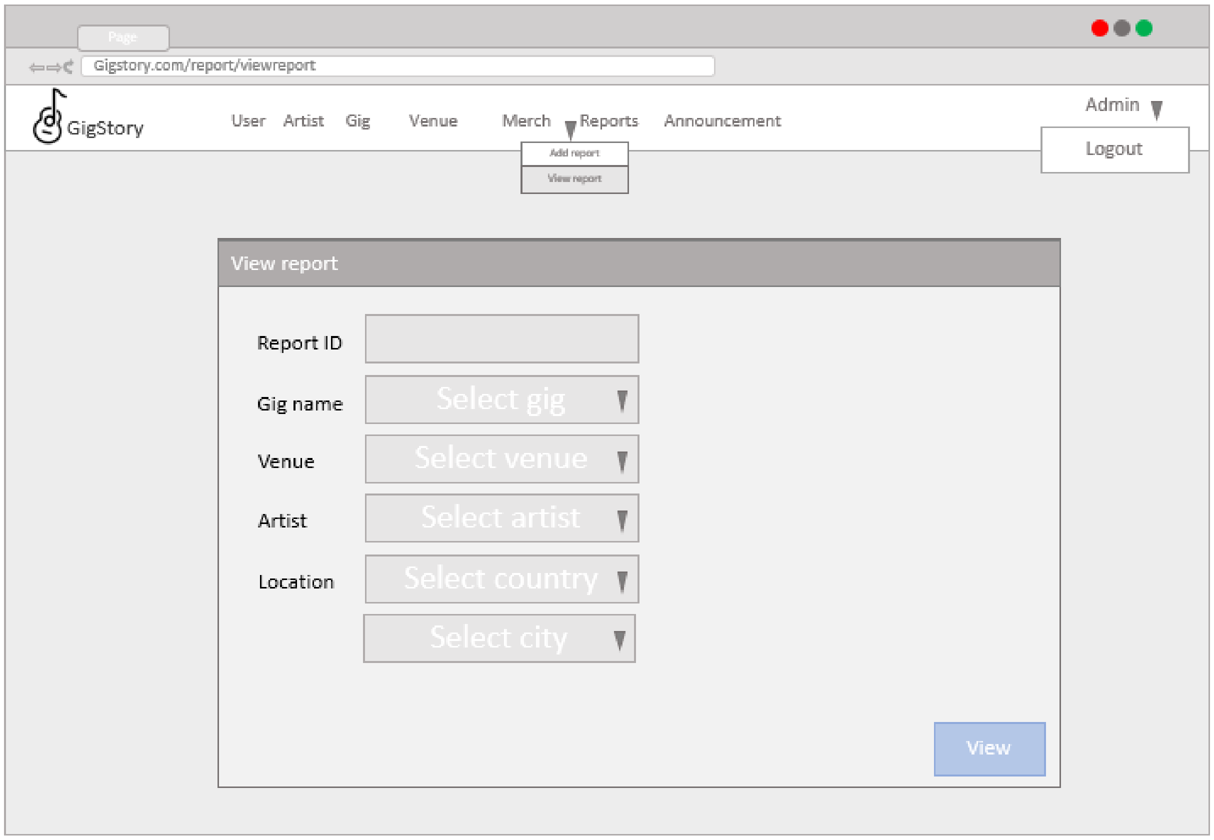
Pic. 40. Add sold merchandise (admin)

On pic. 41 the system should allow admin users to add a new report. The admin user can select the information on what to base the report, like Gig name, venue, artist, location, day of the week, date range, fee range, and also select the information they want to include in the report, like the Total income from tickets, venue, venue price, venue capacity, tickets sold, etc. Based on functional requirements in section 2.2.9, table 34.



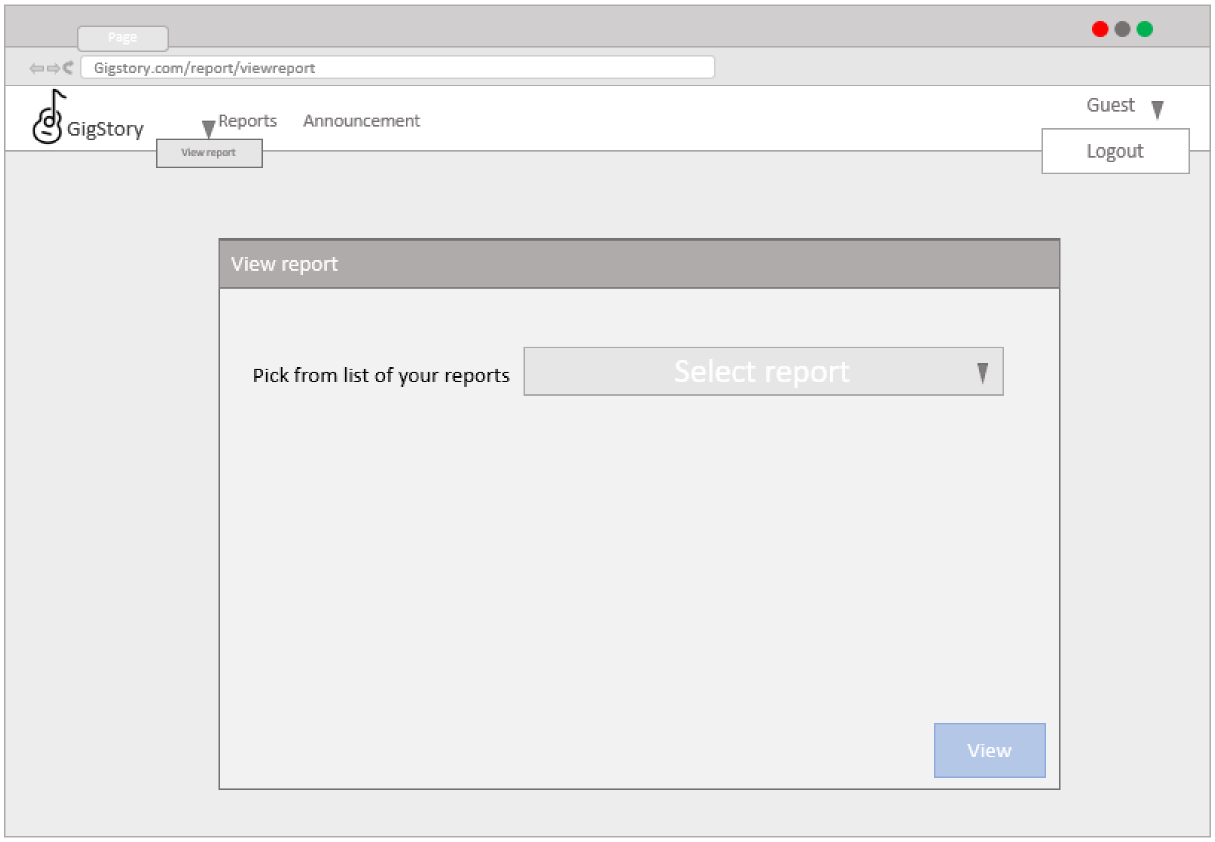
Pic. 41. Add report (admin)

On pic. 42 the system should allow admin users to view reports. The needed reports can be selected by either or all fields - ID, Gig Name, Venue, Artist, Location. Based on functional requirements in section 2.2.9, table 35.



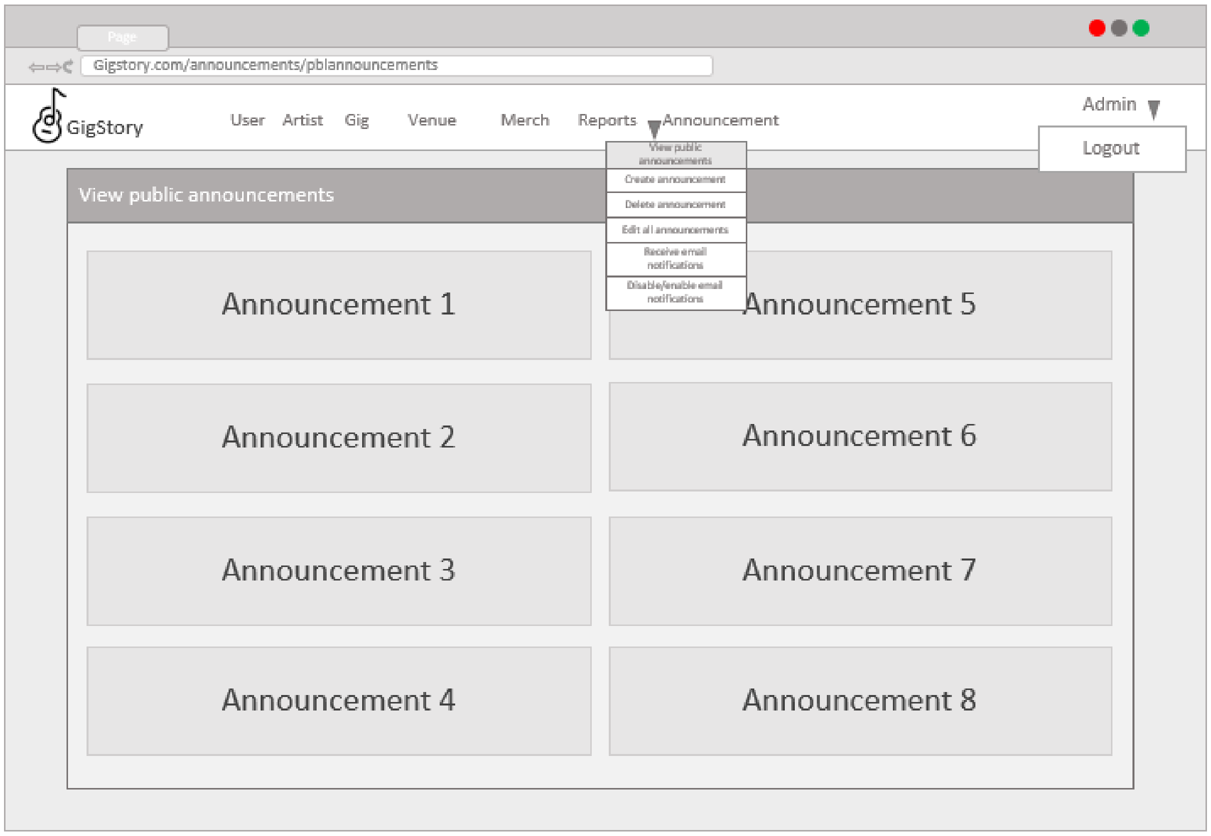
Pic. 42. View report (admin)

On pic. 43 the system should allow guest users to view specific reports that have been shared with them. Based on functional requirements in section 2.2.9, table 35.



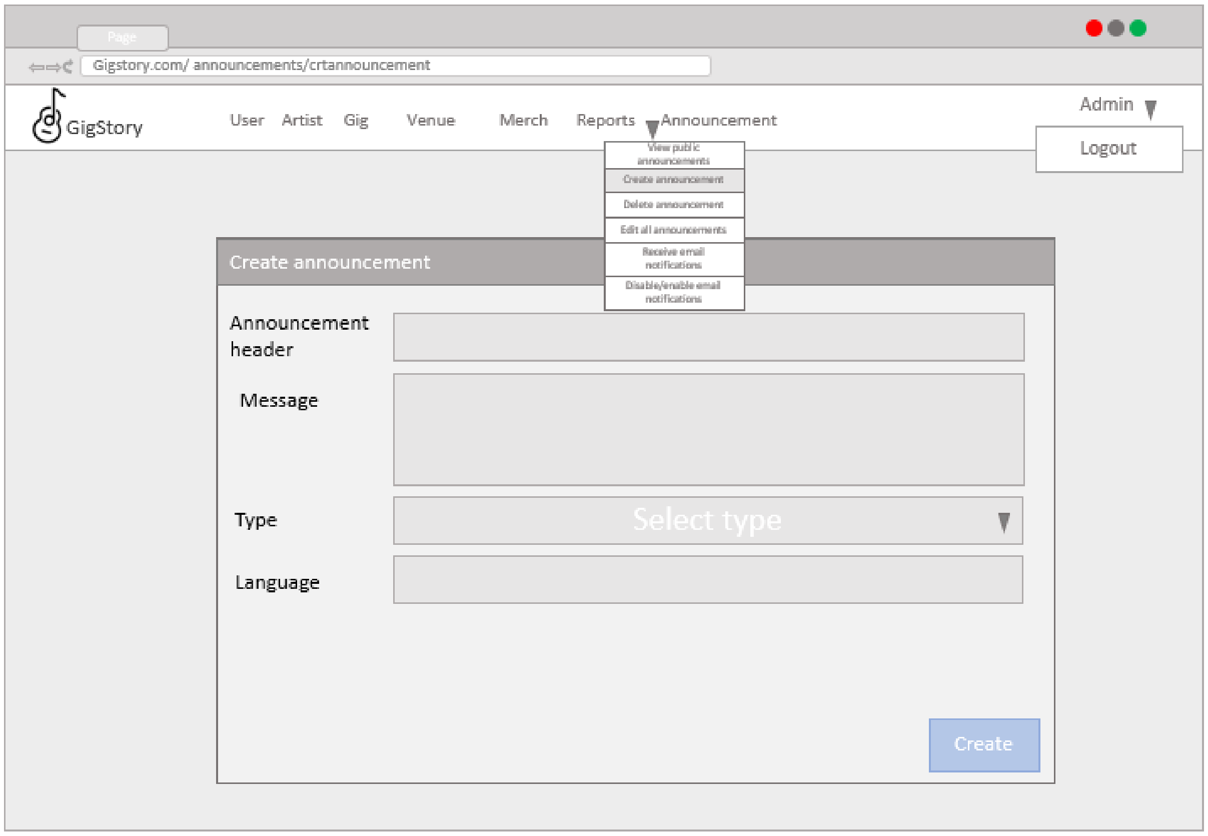
Pic. 43. View report (Guest)

On pic. 44 the system should allow admin users to view all the public announcements in GigStory database. Based on functional requirements in section 2.2.6, table 21.



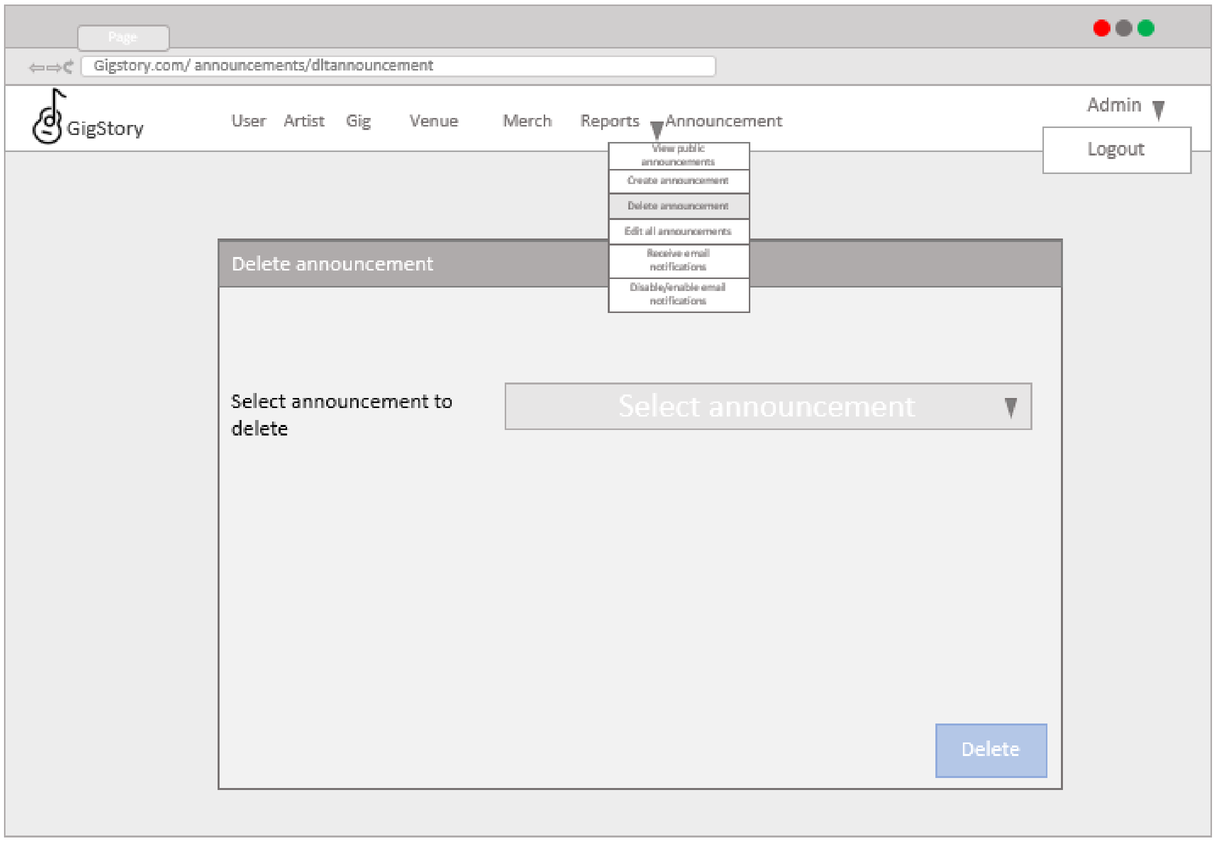
Pic. 44. View public announcements

On pic. 45 the system should allow admin users to create new announcements. The input needed is the header of the announcement, the message, announcement type, and language. Based on functional requirements in section 2.2.6, table 22.



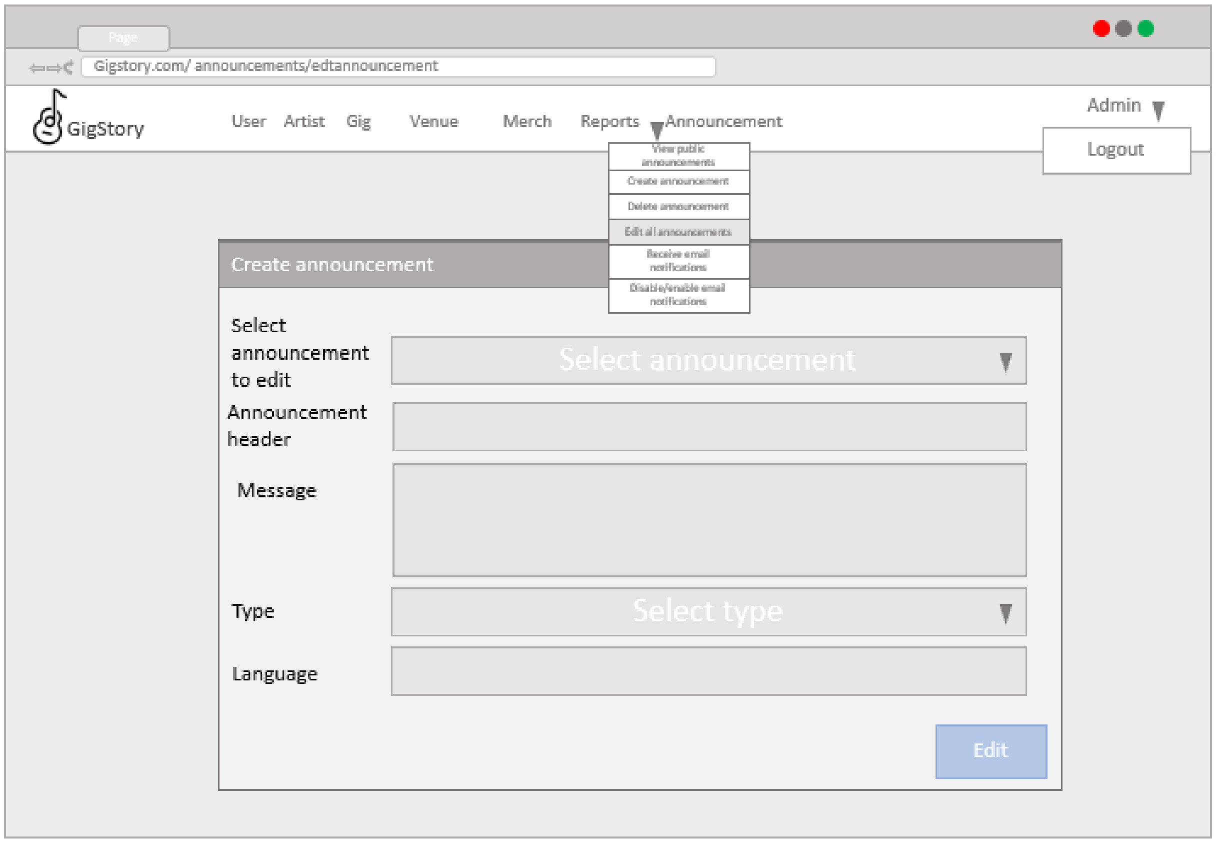
Pic. 45. Create announcement (admin)

On pic. 46 the system should allow admin users to delete any announcement. Based on functional requirements in section 2.2.6, table 23.



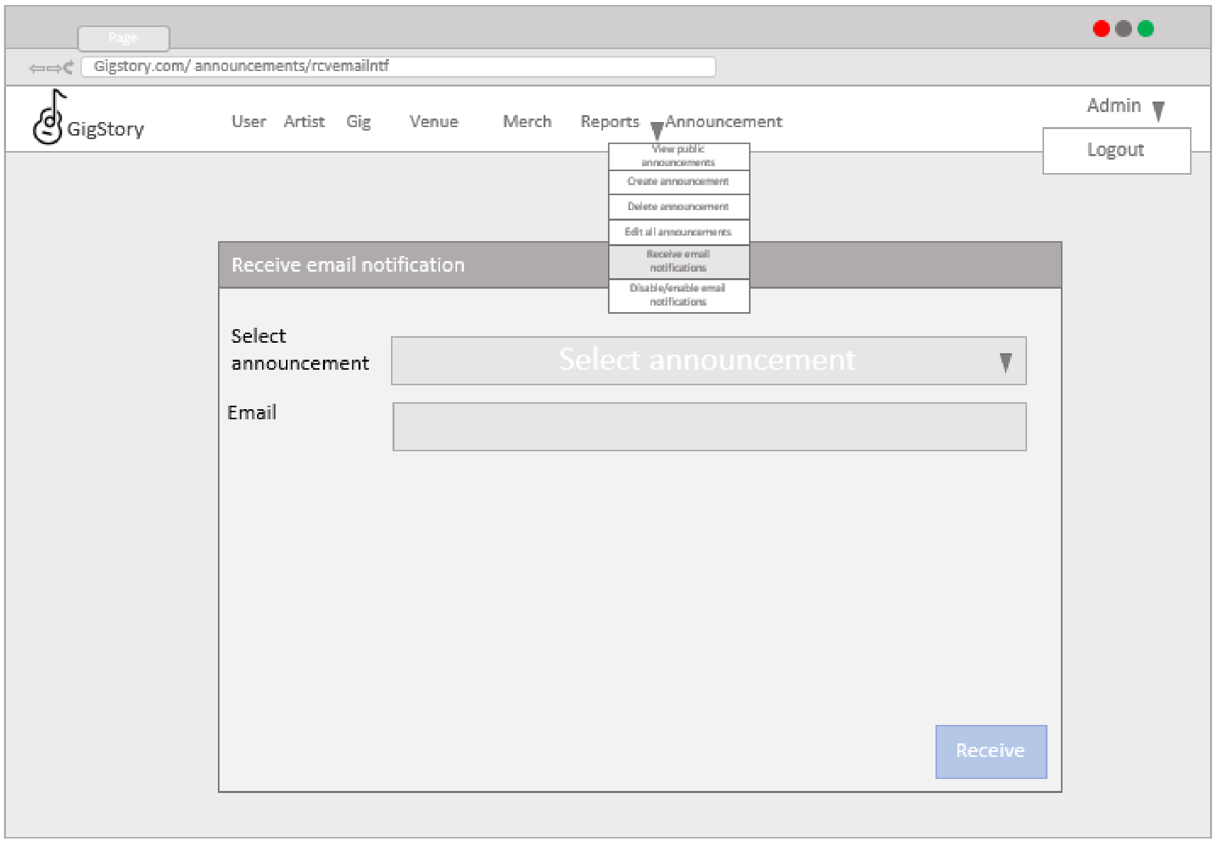
Pic. 46. Delete announcement (admin)

On pic. 47 the system should allow admin users to edit any announcement. Based on functional requirements in section 2.2.6, table 24.



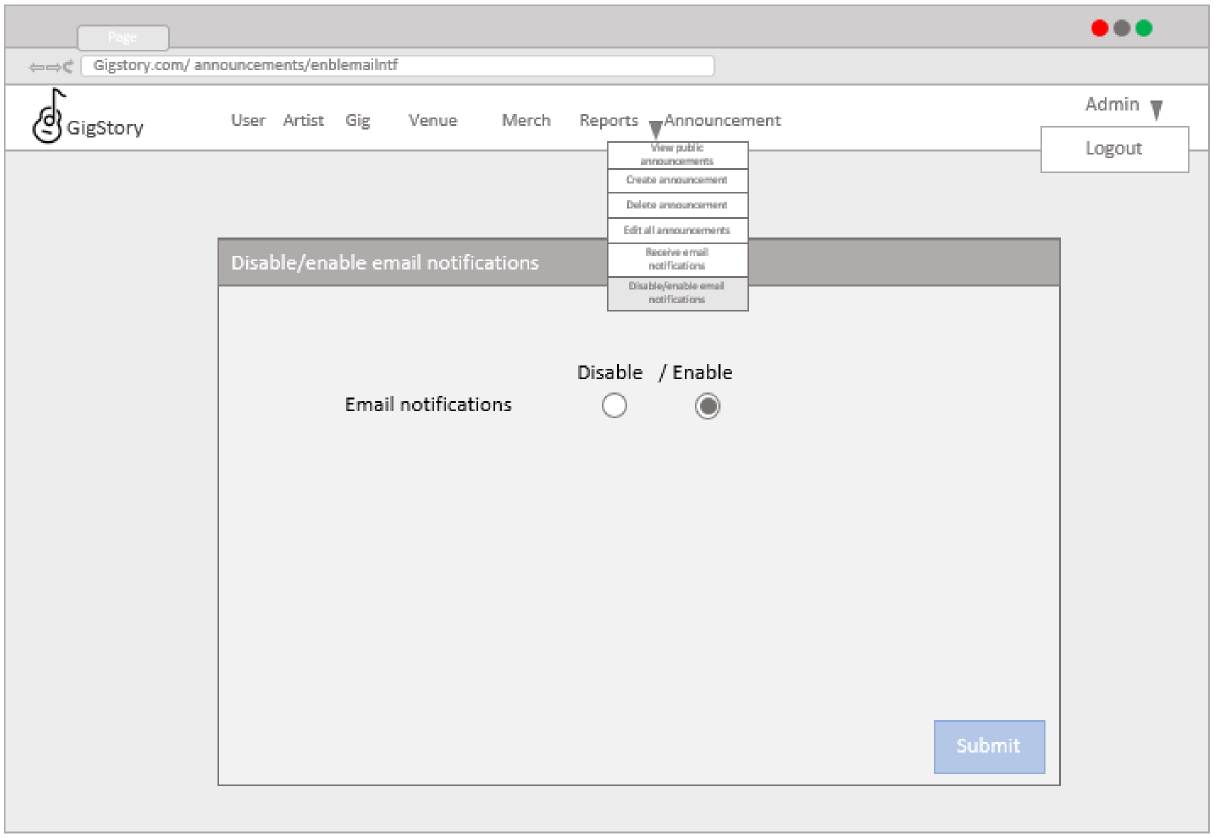
Pic. 47. Edit announcement (admin)

On pic. 48 the system should allow admin users to receive any announcement to any email. Based on functional requirements in section 2.2.6, table 25.



Pic. 48. Receive email notifications (admin)

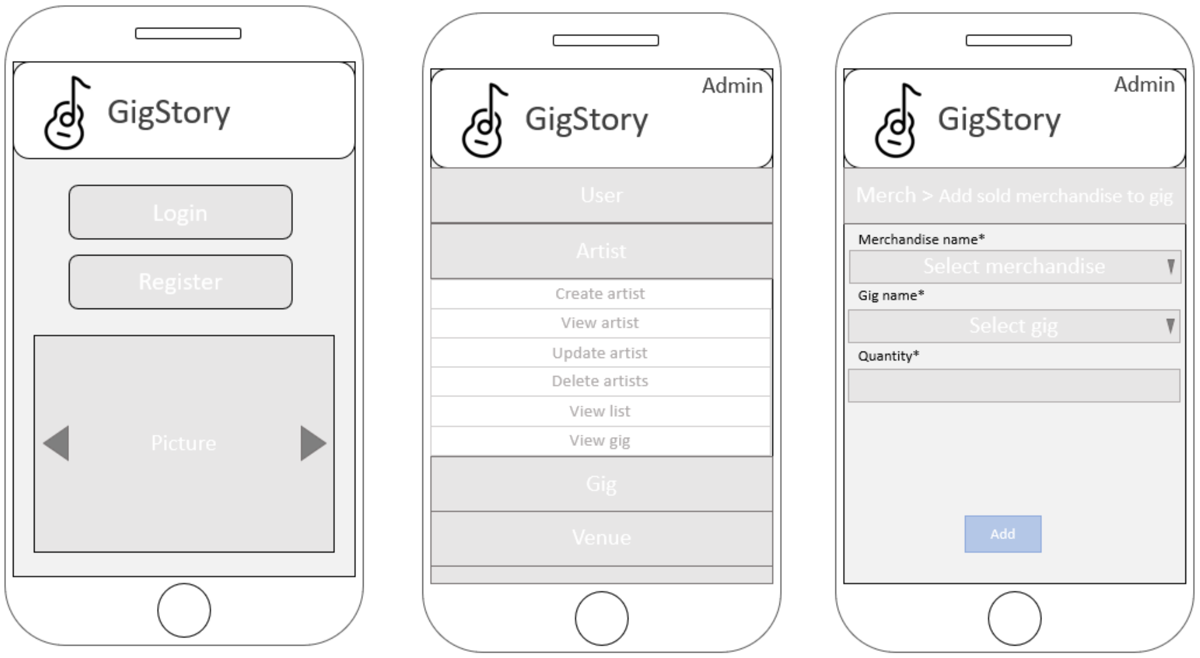
On pic. 49 the system should allow admin users to enable or disable email notifications from GigStory. Based on functional requirements in section 2.2.6, table 26.


Page
Gigstory.com/ announcements/enblemailntf
GigStory
User
Artist
Gig
Venue
Merch
Reports
Announcement
Admin
Logout
View public announcements
Create announcement
Delete announcement
Edit all announcements
Receive email notifications
Disable/enable email notifications
Disable/enable email notifications
Submit

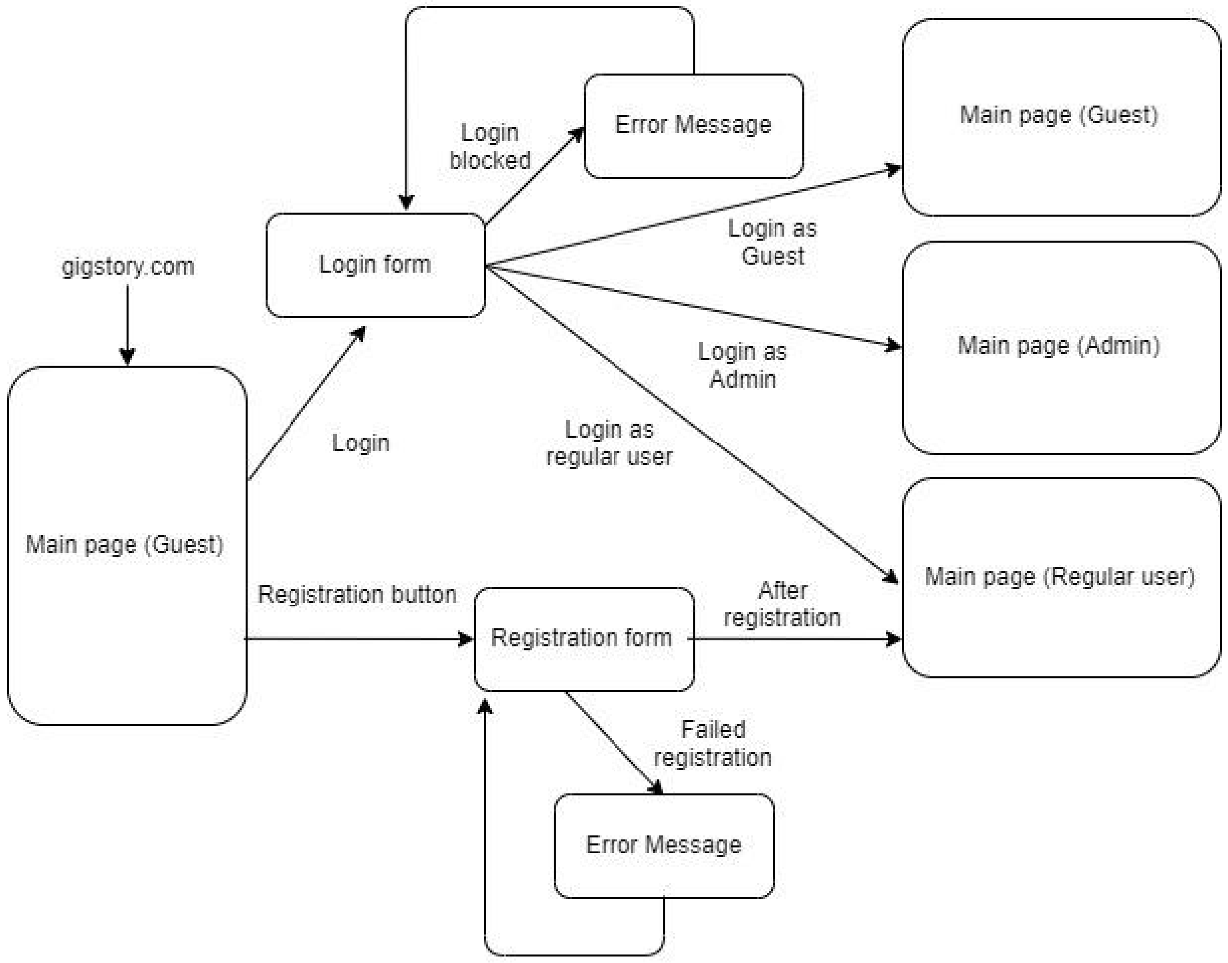
Email notifications Disable / Enable [✓ Enable]

Pic. 49. Disable/enable email notifications (admin)

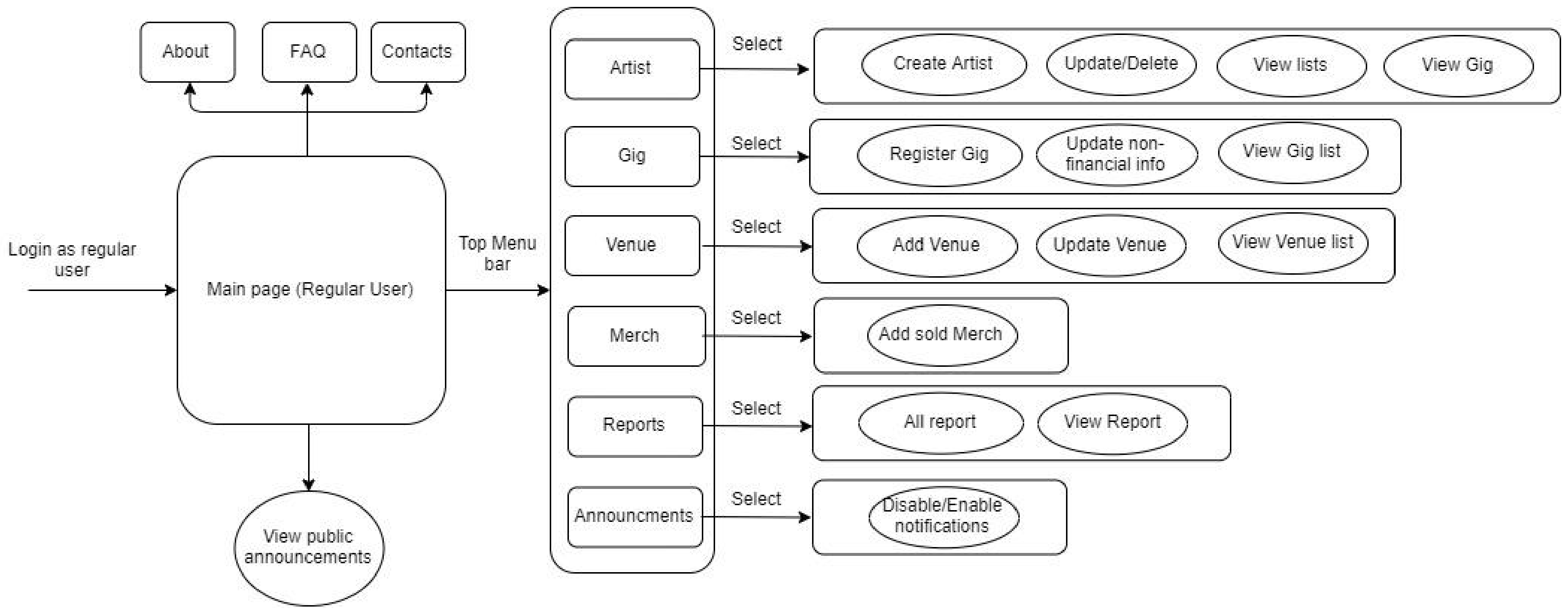
Pic. 50 is a few screen examples of the mobile app that comes with the web application because GigStory will be accessible from both. All of the functionalities will be available on both - web and mobile, but the tabs will be in accordion view on mobile rather than horizontally aligned as in the web version. All of the views will be stacked to reduce horizontal scrolling and make it easy to navigate, the screen will fit users' mobile phone screen width automatically. The style and color scheme will remain the same as in the web version.



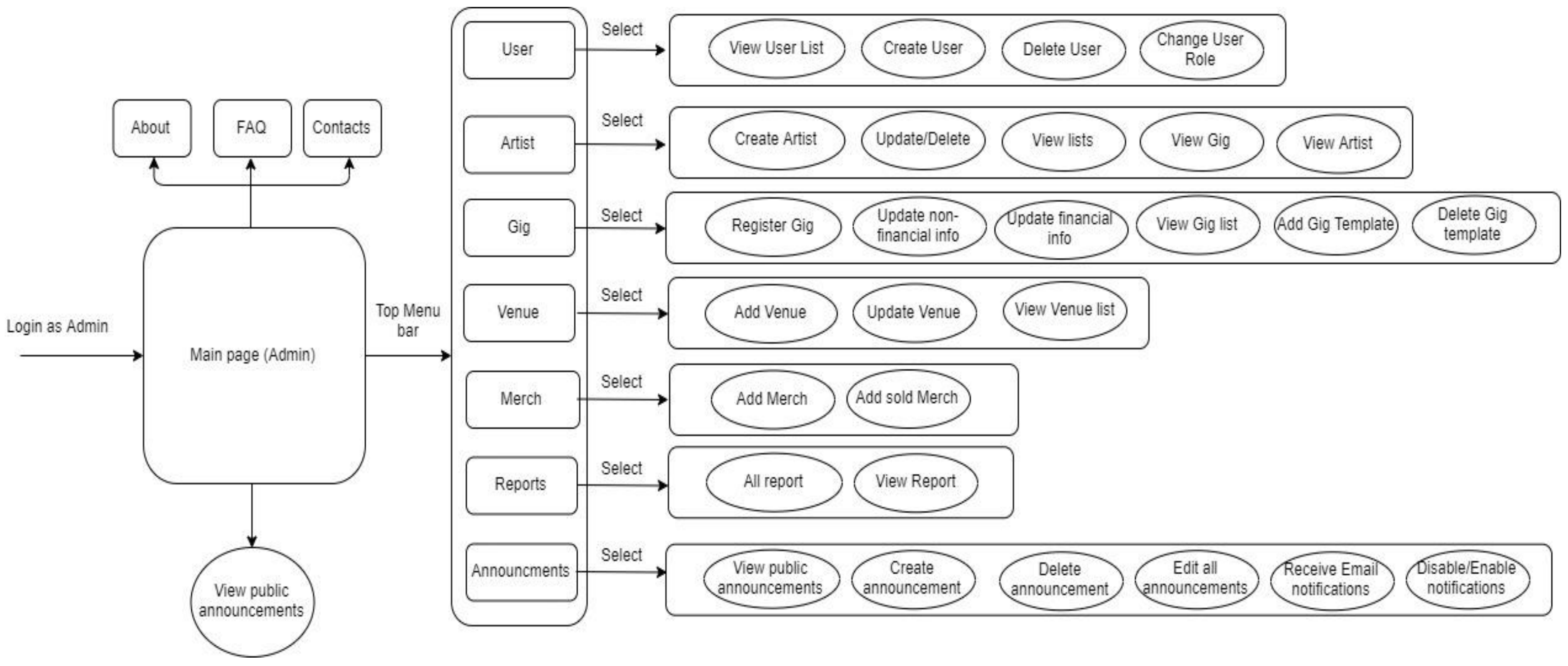
Pic. 50. First page, menu, and "add sold merchandise" views for mobile



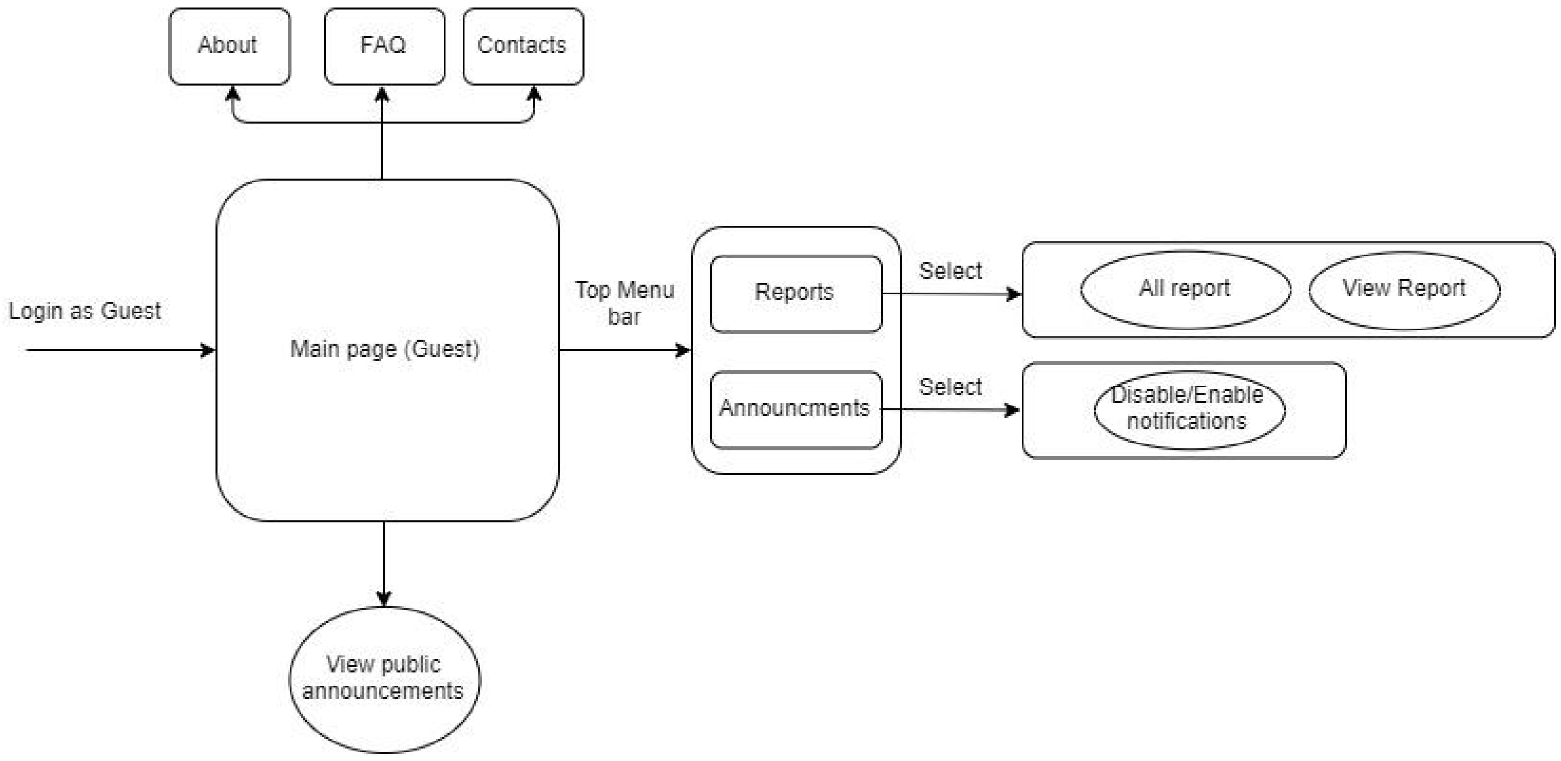
Pic. 51. **User interface navigation diagram (overall)**



Pic. 52. User interface diagram (Regular User)



Pic. 53. User interface navigation diagram (Admin)

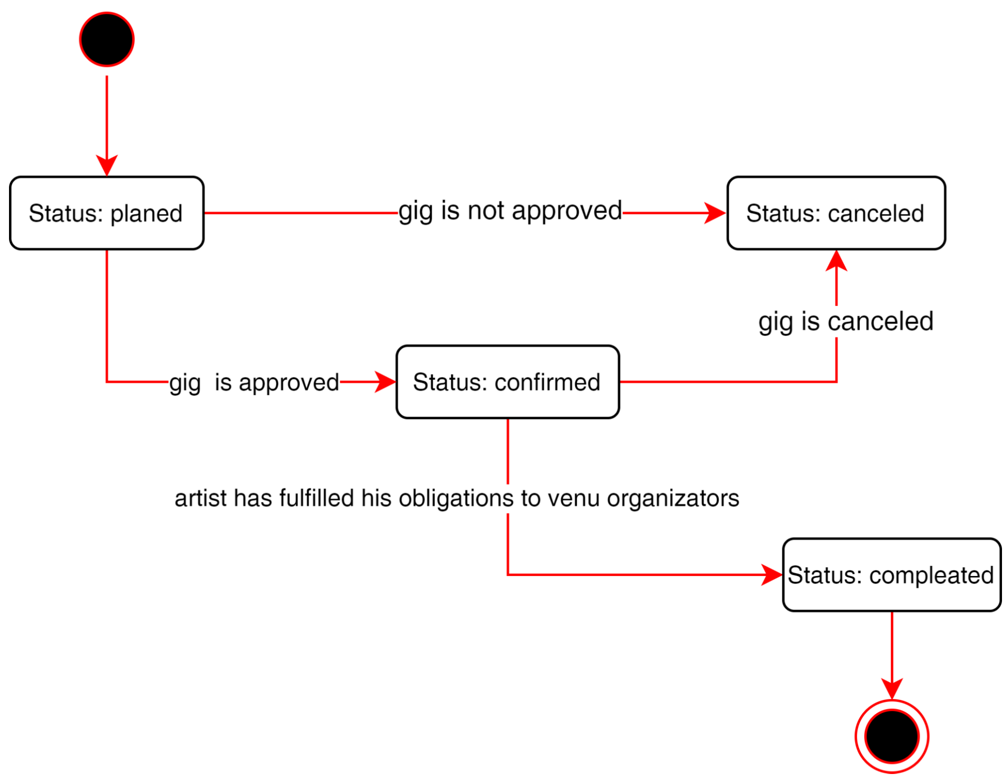


Pic. 54. User interface navigation diagram (Guest)

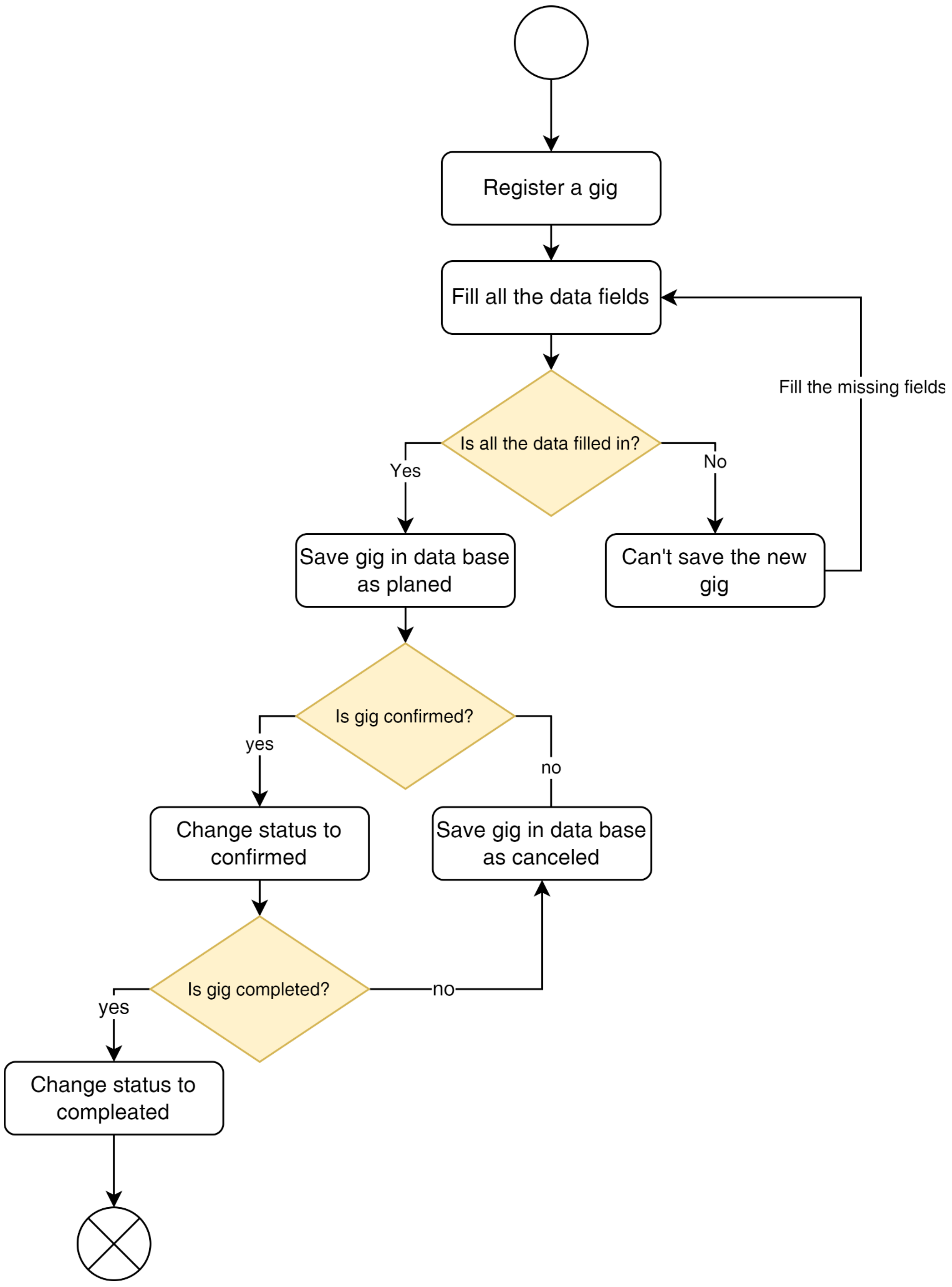
3.3. Partial Functional Design

The section contains diagrams, which describes the most important part of the functionality of the system.

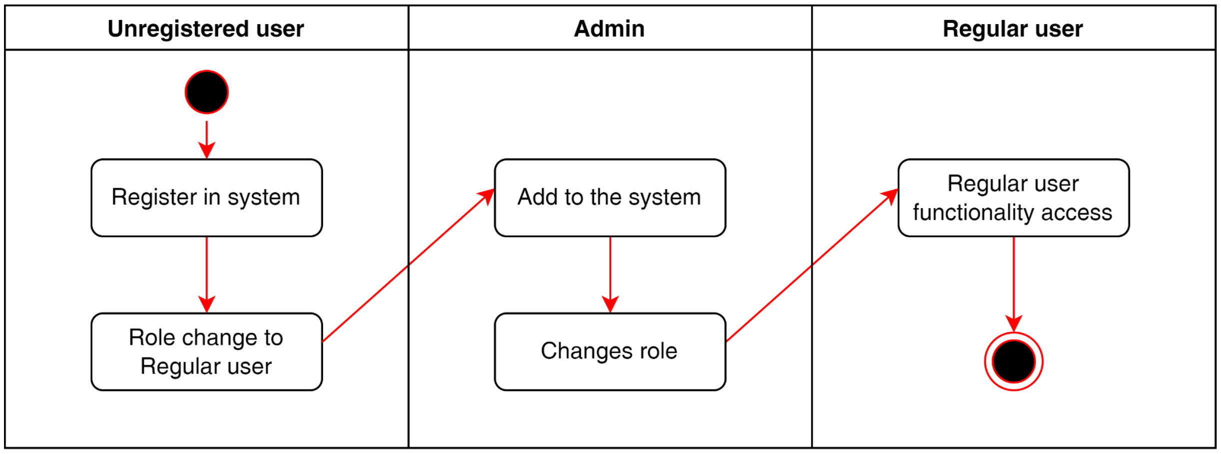
- Gig statuses transition is shown in the pic. 55 which is based on table 40 (Gig entity -description) from section 3.1.3. - Gigs and corresponding entities.
- The process of creating a new gig in the pic. 56 which is based on table 40 (Gig entity -description) from section 3.1.3. - Gigs and corresponding entities and pic. 29 of Partial Design of User Interfaces section.
- The process of configuring an account for Regular users is shown in pic. 57 which is based on table 2 (Registration module) and table 7 (Change user role function module) from section 2.2.3. - User management.
- The process of creating a new Gig is shown in the pic. 58 based on table 14 (register a new gig function module) from section 2.2.5. - Gig management and pic 29 (register new gig (admin view)) from section 3.2.
- The main parts of the system are shown in the pic. 59. It combines all sections described in the document: functional description (sections 2.2.2.-2.2.9.) as controllers, database structure (section 3.1.) in form of models and GUI, described in the section 3.2. as views.
- Class description of a structure in the form of MVC approach is shown in the pic. 60. The centerpiece in the diagram is class Report (Controller) which uses Gig model class for querying the database. In the Bill controller are meant functions shown in pic. 11 in the section 2.2.9. Model class is the corresponding database entity from section 3.13., table 40.



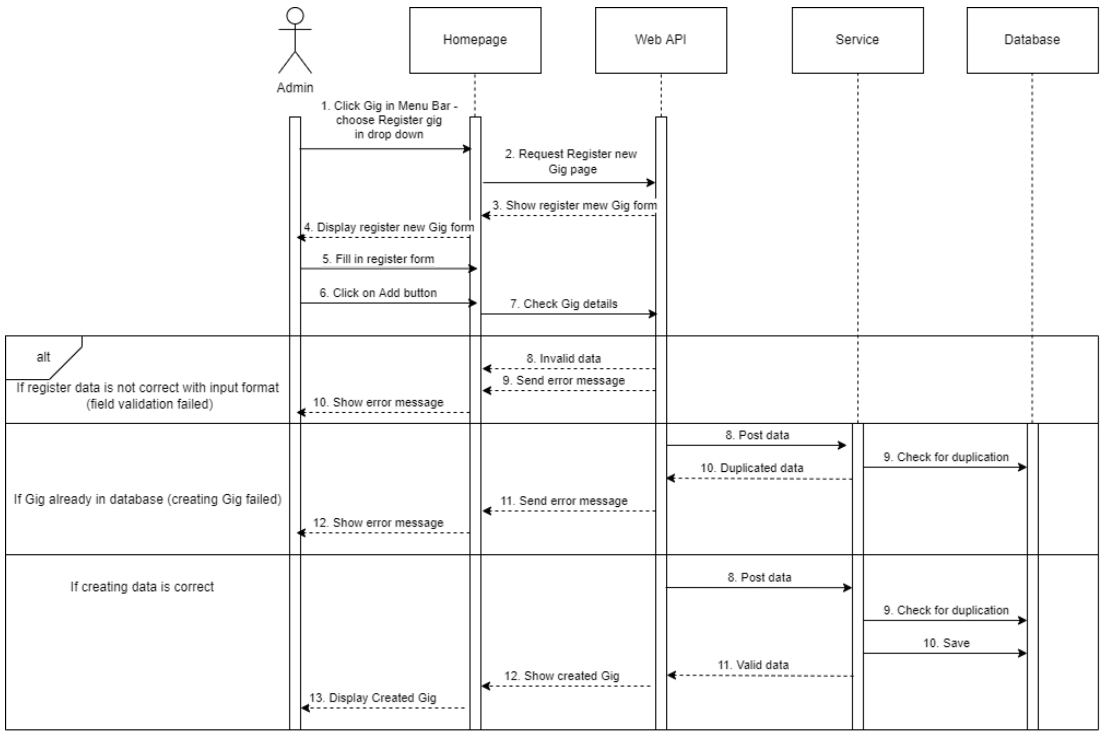
Pic. 55. Gig statuses (UML state transition diagram)


Register a gig
Fill all the data fields
Fill the missing fields
Is all the data filled in?
Yes
No
Save gig in data base as planed
Can't save the new gig
Is gig confirmed?
yes
no
Change status to confirmed
Save gig in data base as canceled
Is gig completed?
yes
no
Change status to compleated

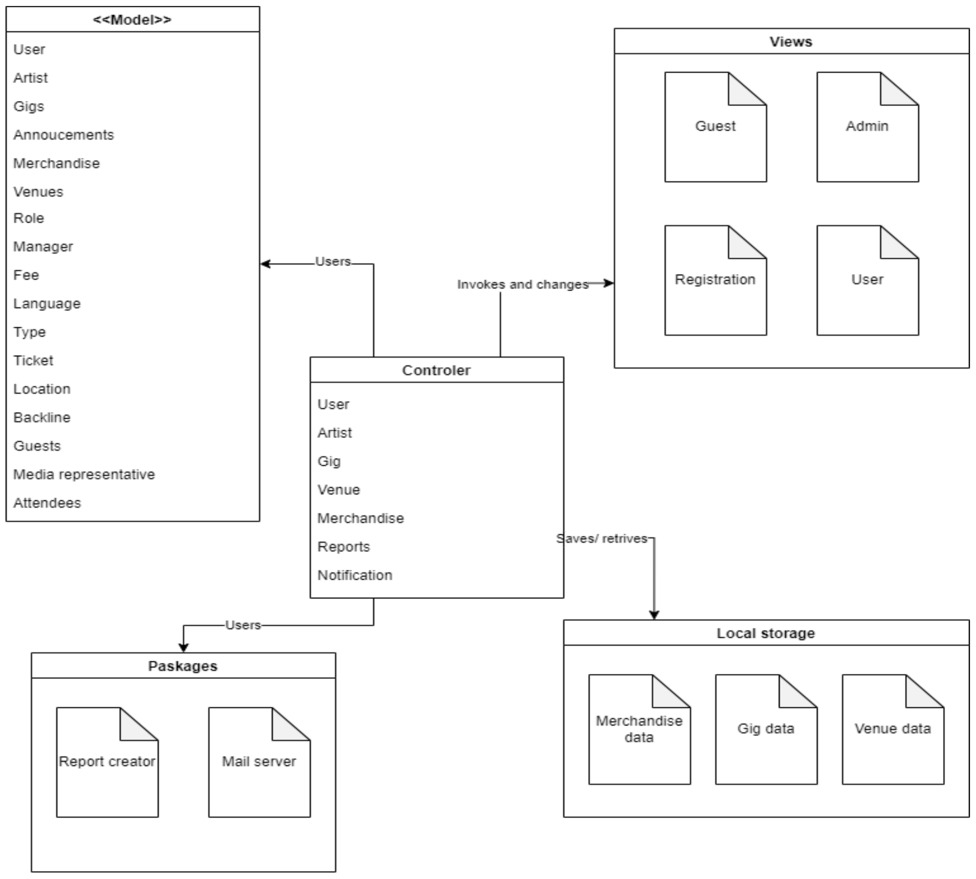
Pic. 56. Register new gig (Block schema)



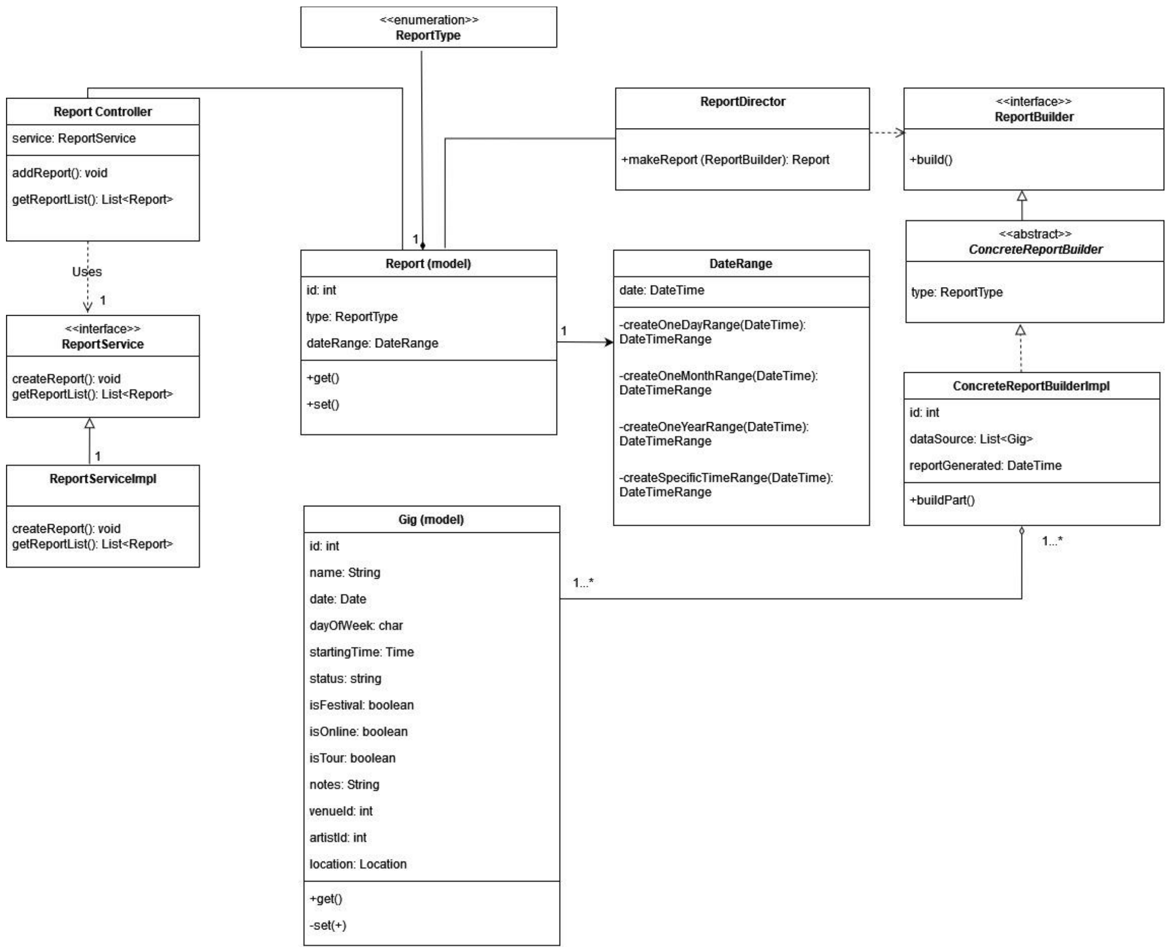
Pic. 57. Configuring working account for Regular user (UML activity diagram)



Pic. 58 Creating new gig (UML sequence diagram)



Pic. 59 Main parts of system (UML package diagram)



Pic. 60 Report Controller - MVC approach (UML class diagram)