ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report

Version 1

CENG 407

Innovative System Design and Development I

P201913

RAD.IO - Online Radio Application for Mobile and Web Platforms

Omar DAYYA
201611504
Mehmet BİLGİÇ
201611009
Berkay KARADAYI
201511034
A. Talha AYDIN
201611006

Advisor: *Roya CHOUPANI*
Co-Advisor*: Murat SARAN*

Table of Contents

Abstract

The internet usage has grown rapidly over the years and radio stations wanted to keep and increase it is user base so, online radios became a thing. This new medium comes with new advantages, such as analyzing the listening audience's demographics; active user count, age, gender, geolocation, behavior (is it previous user or a new user), user agent info, music taste and so on. Data like this allows online radio to give much more quality content, targeted advertising. By this way, online radio attracts more users and keeps the current user.And we want to develop radio application so users will be able to listen to their favourite radio musics.Our application name is Rad.io and we will launch it both web and mobile platform.We will use Flutter to develop the mobile app UI and Django for the web platform as our back-end and front-end we will use HTML, CSS, Bootstrap and other frameworks & libraries needed for the design.

Key Words

Online Radio, Web Platform, Mobile Application, Live Stream, Live Chat, Flutter, Django, Python, Android Studio, Xcode, Bootstrap, Materialize CSS, PostgreSQL, Docker, Google Cloud, Nginx.

Özet

İnternet kullanımı yıllar içerisinde hızla büyüdü ve radyo istasyonları bu büyümeyi korumak ve artırmak istediler.Bu yüzden de online radyolar bir ün haline geldi. Bu yeni ortam, dinleyicilerin demografik özelliklerini analiz etmek; aktif kullanıcı sayısı, yaş, cinsiyet, coğrafi konum, davranış (önceki kullanıcı mı yoksa yeni kullanıcı mı), kullanıcı aracısı bilgisi, müzik zevki vb. Bunun gibi veriler çevrimiçi radyonun çok daha kaliteli içerik ve hedefli reklam sunmasına olanak tanıyor. Bu şekilde, çevrimiçi radyo daha fazla kullanıcı çeker ve mevcut kullanıcıyı tutar. Ve biz de bu radyo uygulamasını geliştirmek istiyoruz, böylece kullanıcılar en sevdikleri radyo müziklerini dinleyebilecekler.Uygulama adımız “Rad.io” ve hem mobil hem de web platformu için çıkaracağız.Mobil uygulama arayüzü

geliştirmek için Flutter'ı, web platform arayüzü için Django'yu back-end; HTML, CSS, Bootstrap, framework ve tasarım için gerekli olan diğer kütüphaneleri de front-end olarak kullanacağız.

Anahtar Kelimeler

Çevrimiçi Radyo, Web Platformu, Mobil Uygulama, Canlı Yayın, Canlı Mesajlaşma, Flutter, Django, Python, Android Studio, Xcode, Bootstrap, Materialize CSS, PostgreSQL, Docker, Google Cloud, Nginx.

1. Introduction

Internet radio (also web radio, net radio, streaming radio, e-radio, IP radio, online radio) is a digital audio service transmitted via the Internet. Broadcasting on the Internet is usually referred to as webcasting since it is not transmitted broadly through wireless means. It can either be used as a stand-alone device running through the internet, or as a software running through a single computer [1]. Internet radio is being evolved day by day, new features are being added and new researches have been developed. It is these advancements in wireless telephony and other devices that led us to radio becoming the broadcasting medium we enjoy today. Radio as a broadcasting medium brought with it important cultural consequences that must be considered when examining the future of radio in today's context.

The internet radio involves a streaming media, presenting listeners with a dynamic stream of audio that cannot be paused or replayed, much like FM/AM radios that are found in cars. Internet radio includes both mobile application and web application. In these days, mobile applications are having a better advantage in comparison to website radio streaming services.

1.1 Problem Statement

Listening to online radio stations is easy due to development in technology.You can listen to radio using your phone or a physical radio device.But in online radios sometimes there are serious problems that can prevent users to listen to radio. Since we are working on online, our biggest problems are delays in stream or server downtime.

1.1.1 Stream Delays

Stream delay is synchronization problem between streamer and stream. If sound outs from the source later than streamer's speech, it is a delay. In radio and television, broadcast delay is an intentional delay when broadcasting live material. Such a delay may be short (often seven seconds) to prevent mistakes or unacceptable content from being broadcast. Longer delays lasting several hours can also be introduced so that the material is aired at a later scheduled time (such as the prime time hours) to maximize viewership. Tape delays lasting several hours can also be edited down to remove filler material or to trim a broadcast to the network's desired run time for a broadcast slot, but this isn't always the case [2].

1.1.2 Server Downtime

Downtime refers to periods of time during which a computer system, server or network is shut off or unavailable for use.

1.1.2.1 What causes downtime?

1.1.2.1.1 Human Error

Various studies over the last several years have placed human error as most frequent causes of server downtime. Whether through accident or negligence, many of the highest profile service outages of the last few years can be directly traced back to human error.

1.1.2.1.2 Cyberattack

One of the high-profile downtime causes, cyberattacks usually make big headlines when they occur. Network vulnerabilities create opportunities for hackers to infiltrate systems, allowing them to steal data, shut down applications, and lock out users with ransomware. Even if a system is relatively secure, it may still be vulnerable to a distributed denial of service (DDoS) attack that can paralyze and crash servers that aren't prepared to withstand the traffic spike.

1.1.2.1.3 Equipment Failure

Sometimes equipment just breaks. It's an unpleasant truth, but data center physical infrastructure is always vulnerable to failure of some kind, making it one of the leading causes of downtime.

1.1.2.1.4 Software Failure

Although less common than hardware failures, network systems are only as effective as the software they're running. When operating systems are updated with patches that haven't gone through proper testing, entire applications can become corrupted and bring networks screeching to a halt. However, outdated software is often just as problematic because it lacks current security measures or drivers to keep high traffic networks up and running.[3]

2. Literature Search

2.1 What Is Radio?

Radio is the technology of signaling and communicating using radio waves.In radios(AM/FM) use signals to recognize the particular waves for bradcast.Radio waves are electromagnetic waves of frequency between 30 hertz and 300 gigahertz[4].Electromagnetic radiation travels by means of oscillating electromagnetic fields that pass through the air and the vacuum of space. Information, such as sound, is carried by systematically changing some property of the radiated waves, such as their amplitude, frequency, phase, or pulse width. When radio waves strike an electrical conductor, the oscillating fields induce an alternating current in the conductor. The information in the waves can be extracted and transformed back into its original form [5].

2.2 What Is Online Radio?

Online radio means listening to radio via the internet. The radio signal is not transmitted via AM or FM, but streamed via the internet. This means that your device needs to be connected to the internet to receive the radio station. The internet connection can be Wi-Fi or mobile data.These services are accessible via the internet on lots of devices, including smart TVs, tablets, computers, laptops and smartphones.

Streaming means the content that you are listening to is brought to you directly from the internet. If your internet is not fast enough then the radio station will start and stop playing. This is called buffering [6].

3. Summary

3.1 Technology Used

We propose a web and mobile application. Our mobile part will use Flutter and backend will use Django framework which is a high-level Python Web framework that encourages rapid development and clean, pragmatic design, and PostgreSQL as relational database management system, and we will deploy our application behind Nginx in cloud platform.

3.2 Mobile Platform

The radio application for the mobile platform will include mobile services as well as a mobile application development framework for the UI which will be programmed using Dart. The framework used to develop the mobile app UI is Flutter.

3.2.1 What is Flutter?

Flutter is a cross-platform mobile application development framework developed by Google that can be used for iOS and Android and is programmed using Dart programming language.

To develop Flutter applications, you will need an IDE apparently, some examples for Flutter IDEs are Visual Studio Code, IntelliJ, Android Studio, etc.. Such IDEs work on developing Android applications only, unfortunately, as for iOS development, we will need to have a Macintosh device to be able to develop a mobile application for iOS. iOS devices need Xcode IDE which is only available on Mac devices. In addition, other tools will be needed like Xcode simulator and Android emulator to display the UI of the mobile applications.

3.3 Web Platform

The radio application for web platform will include web services to allow radio broadcasting among listeners as well as a web application web page which will be coded using Django as our back-end and as for the front-end we will use HTML, CSS, Bootstrap and other frameworks & libraries needed for the design.

3.3.1 What is Django?

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site [7].

4. Software Requirements Specification

4.1 Introduction

4.1.1 Purpose

This document provides detailed information about the requirements of the radio system software. It will include the vision statement and domain analysis regarding the project. As well as designs and detailed description of the system for both mobile and web platforms. Our intended audience is streamer and listeners.

4.1.2 Scope of Project

Nowadays, radio becomes less popular because of some applications like Spotify. Users can reach any song at anytime and they listen to people instead of radio streamers from applications like Youtube and Twitch. We think radio concept needs innovation to keep up with the times. For this reason, we aimed to develop a fast, interactive, user-friendly and innovative online radio application for both web and mobile platforms which allows users and listeners more interactivity.

The purpose of Rad.io project is to design a “Online Radio Application for Mobile and Web Platforms“ which gives users more socialization chance with different additions. Our platform will use modern solutions and functionalities such as analyzing the audience and improving our user experience based on it. Minimalist and responsive design to improve usability. Scalable server infrastructure to handle the incoming traffic using modern solutions.

There are new points of view for a radio application. We aimed to combine radio concept with new ideas which are suitable for the internet age. The Rad.io system provides people a basic

and functional online radio broadcasting station with a convenient UI and new things to learn and fun timings with others.

4.1.3 Glossary

Table 1 Glossary of SRS

Term	**Definition**
SAM 2	SAM Broadcaster is an internet radio broadcasting application. The name 'SAM' is an acronym for Streaming Audio Manager, which describes the software's functionality.
Streamer / Broadcaster	Person who runs the stream. Admin.
Spotify	A digital music, podcast, and video streaming service that gives you access to millions of songs and other content from artists all over the world. [1]
User	Person who uses the application for listening radio.
UI	User interface.
Android	A mobile operating system.
iOS	A mobile operating system for Apple devices.

4.1.4 References

[1] https://support.spotify.com/us/using_spotify/getting_started/what-is-spotify/

4.1.5 Overview of Document

In first part, purpose and scope described. The second part of the document describes functionalities and the third part describes requirement specifications of the Project Rad.io: Online Radio Application for Mobile and Web Platforms. Both parts describe the same application, but they include different explanations due to different audiences. The second part is written for general and non-technical explanations. The third part is written for developers with technical and detailed information.

4.2 Overall Description

4.2.1 Product Perspective

Project Rad.io is an internet radio application which has a new perspective for internet radio. Both mobile and computer users access the platform with an internet connection. For keep connecting with new technology and platforms, there is Spotify registration possibility. There are beneficial features for both streamer and users. Streamer can ask questions anytime and see answers at the same time. Users can add songs to their Spotify playlist if they like it or like the song for new song recommendations. With this new features and perspective, radio concept will become popular again.

4.2.2 Product Functions

This part includes major functions of the Project Rad.io with non-technical explanations.

4.2.2.1 Live Radio Broadcast

The main purpose of the project is live radio broadcast. It allows users to listen live radio broadcasting.

4.2.2.2 Non-Live Radio Broadcast

Users able to listen music from playlist prepared before by admin when there is no live broadcast.

4.2.2.3 Membership

There are two membership opportunities: Normal registration and Spotify registration.

4.2.2.4 Live Chat

All users able to read the chat, but just registered users can write to the chat.

4.2.2.5 Song Recommendation

Registered users able to see song recommendations on their profiles according to their likes.

4.2.2.6 Live Survey

Users can vote for streamer's instant surveys.

4.2.2.7 Adding Songs to Spotify Playlist

Users, which registered with Spotify account, can add playing songs to their playlist.

4.2.2.8 User Ban

Admin able to ban users.

4.2.2.9 Events and News

Admin able to add news or events for users. Users can read this news from new page.

4.2.2.10 Podcasts

Users are able to listen to old records of streams.

4.2.2.11 Voice Chat (Optional)

Streamer able to invite any member from live chat to voice chat for a conversation.

4.2.2.12 Event Recommendation (Optional)

Application recommend events according to user's location with notifications on mobile platforms.

4.2.3 User Characteristics

4.2.3.1 Streamer/Admin

- Admin should know how to use computer for broadcast.

- Admin should know how to use mixer software (SAM 2).

4.2.3.2 Unregistered User

- Unregistered user should know how to use computer or smartphone with Android or iOS.

4.2.3.3 Registered User

- Registered user should know how to use computer or smartphone with Android or iOS.

4.2.3.4 Spotify-Registered User

- -Spotify-Registered user should know how to use computer or smartphone with Android or iOS.

- Spotify-Registered user must have a Spotify account.

4.2.4 Constraints

Rad.io is limited to the data, it will use the internet.Also, storage capacity of the device as it takes up space in the device to be used.

4.2.5 Risks

- High network traffic.

- Connectivity issues about broadcast. If streamer disconnects as b plan player will play automatic music.

- Cross-platform incompatibilities. Some functions may not work on other platform, as b plan we may use wrapper for platform specific features.

4.3 Requirements Specification

4.3.1 External Interface Requirements

4.3.1.1 User interfaces

The user interface will be worked on Android, iOS and web browsers (Google Chrome, Firefox etc.).

4.3.1.2 Hardware interfaces

A computer and microphone for streamer

Users must have a computer or smartphone with Android or iOS.

4.3.1.3 Software interfaces

Streamer must have SAM 2 software on his/her computer.

4.3.1.4 Communications interfaces

Internet connection for both admin and users.

4.3.2 Performance Requirements

The minimum system requirements for smartphones:

Operating System: Android 4.4 or better - iOS 10 or better.

Storage: 100 - 200 MB

Ram: 1 GB or better

4.3.3 Software system attributes

4.3.3.1 Performance

Good internet connections may give better results and application usually sends audio data, so it will not spend so much data.

4.3.3.2 Availability

Rad.io available for web browsers and mobile platforms. Some functions available just for registered users.

4.3.3.3 Usability

- Users can directly starts the player.
- Users can like the song with one touch.
- Menu is very simple and there are not many different options.
- Users can connect with their Spotify accounts with one button.
- Music continues on background when user change the application.

4.3.3.4 Scalability

We aimed not to encounter a user limitation problem. For now, we can host 500-1000 users at the same time, but it may increase.

4.3.3.5 Portability

Rad.io designed for also mobile platforms (Android and iOS) so it can work with both operating systems.

4.3.3.6 Ease of Use

Project Rad.io is a user-oriented project. It will have a UI that will provide simple usage to user and this UI will be understandable and user- friendly.

4.4 Requirements Specification

4.4.1 Use Cases

4.4.1.1 Actor(s)

User: The user is a person who download the application and uses it. For using the application, user must accept the Cookies once to allow identify users and save site login information. The user can register to the website to become a member of Rad.io to gain more features. The user must accept the terms and conditions and must verify their account through email.

4.4.1.2 Stakeholder(s)

Project Advisor: The project advisor is responsible for delivering the project on time and on a given budget. They usually work together and guide the team that develops the project so that the goals in the project can be fulfilled correctly.

The Project Manager's goals are; To follow the project and check if it is done as requested. To ensure that the risks and problems are properly handled. Working with the group to motivate them and give the necessary support.

Project Development Team: It is a team of people who design the project given by the project advisor according to the desired characteristics and who play an active role in the project by working together within the project.

The Project Development Team's goals are; To make the project timely and correctly. To be able to complete the project in accordance with the quality and rules as much as possible. [3]

User: The user is the person who will use the project after the project is finished. User is the person who gives an idea to the project developer about the needs of the project and be reference in the project development process. The User's Goals are;

- Accessing the desired radio station in a fast way.
- Self entertaining by chatting with other members/radio host.
- Accessing spotify songs and sharing it among other members.

4.4.1.3 Use Case Diagram

Figure 1 presents a use case diagram for the Rad.io application. The system shows the operations that end users can perform.

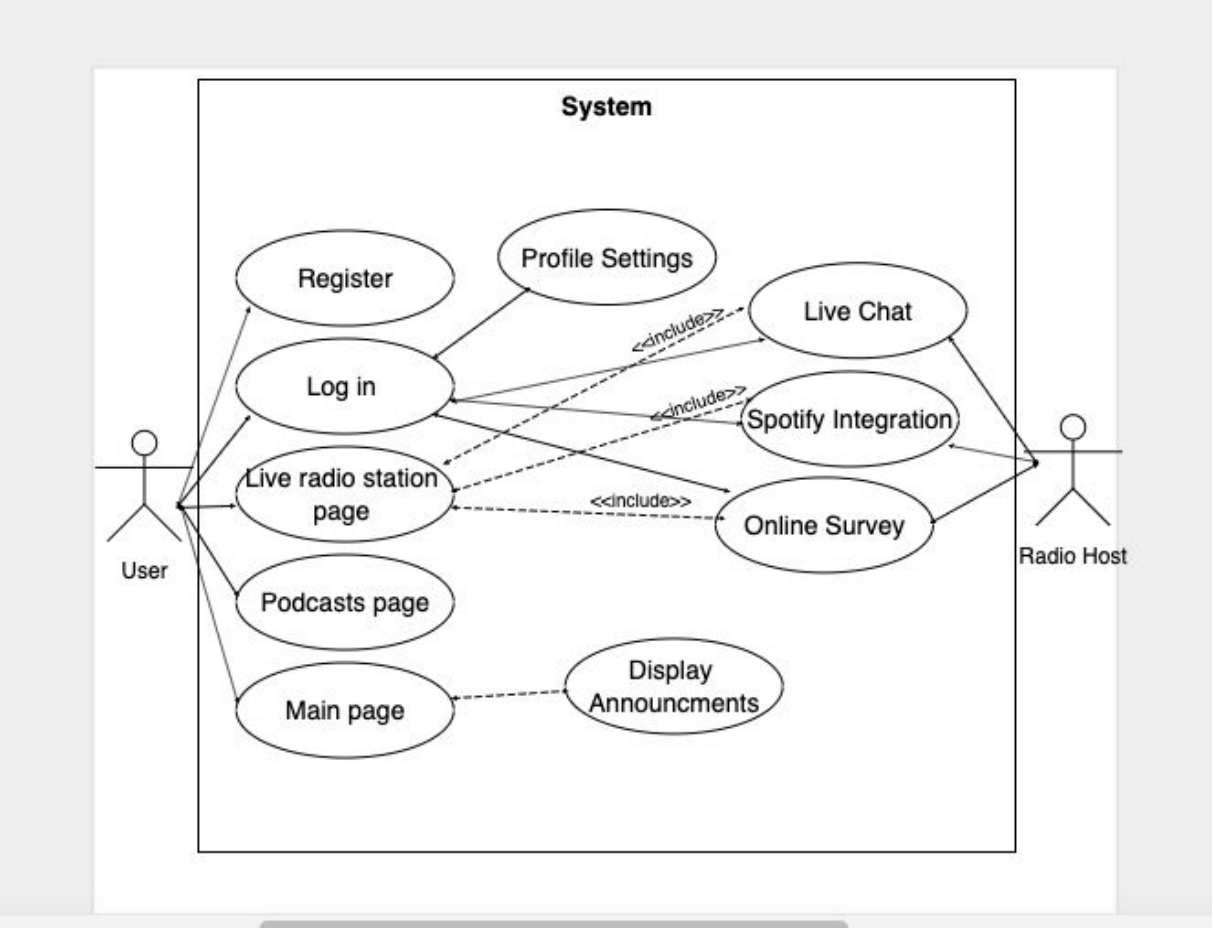


Figure 1 Rad.io Use Case Diagram

4.4.1.4 Brief Description of Use Case Diagram

The use-case diagram shows us the web pages that the user and radio host can access. The user can register to the website to gain more features, non-members can also access the website but cannot have the extra features. As shown, when the user accesses the "Live Radio station page" he/she can see the features but cannot use them unless they are registered to the system. Non-members can read the live chat messages but will not able to chat with other members, they can see surveys but cannot participate, and so on. The radio host has the

opportunity to take control of actions and operations over the live chat, spotify integration, and online survey during live streaming of the station.

4.4.2 Key - Use Cases

4.4.2.1 Membership Use Case

- User Registration
- User Agreement
- Login
- Check Information About User
- Login Failed
- Change Password

Brief Description:

Using this application requires an account.If users don't have an account they can create an account by pressing "Sign in" button.Users who want to use the Rad.io application, has to create an account to use main feature like chatting with other users.

Step-By-Step Description:

1) When users want to use Rad.io app, first they need to download from App Store or Google Play Store. Also, users can access Rad.io from the Web Platform. After entering the app they can listen to broadcasts, view chat but they have to register to Rad.io for more features.

2) In registration part user has to write their information about themselves like username, email, password etc.After writing e-mail , verification about registration will be sent to their email address and user confirms this to continue Rad.io app.

3) After confirmation , users can use the full features of Rad.io app like chatting with other users, adding songs to Spotify account(if they entered with the Spotify registration).

4) After registration, users can log in to the app but if they forget their password, they will be able to change password.

4.4.2.2 Dashboard Use Case

- Announcements
- Survey Result
- Statistics

Brief Description:

In Dashboard, users will be able to see the statistics ,announcements about site, music and time of broadcasts. Also they will be able to know the results of the surveys that they voted or others' vote.

4.4.2.3 Host Use Case

- Login for admin
- Ban User
- Create/Delete Announcement
- Create Survey

Brief Description:

Admin of the system has to log in to make changes like adding/deleting announcements. He/she can arrange the time for broadcast and start/stop it. In broadcast there will be a chat for users. If some of users disrespect(or worse) to other users, admin will be able to ban those users. Also during broadcast or later admin will be able to create survey for songs, etc.

4.4.3 UML Class Diagram

Figure 2 presents a class diagram for the Rad.io application.

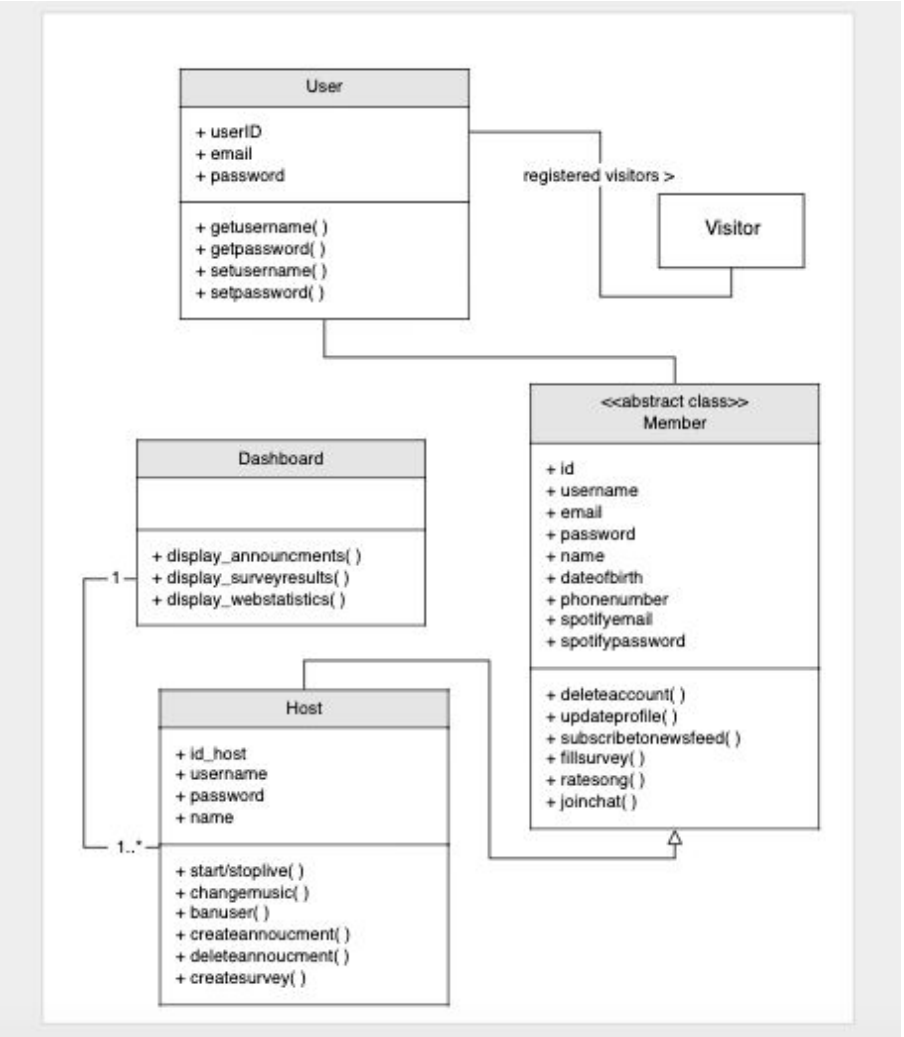


Figure 2 Rad.io Class Diagram

4.4.4 UI Sketches

Figure 3 presents UI sketches for the Rad.io application. Sketches shows the menu and tabs of the project Rad.io.

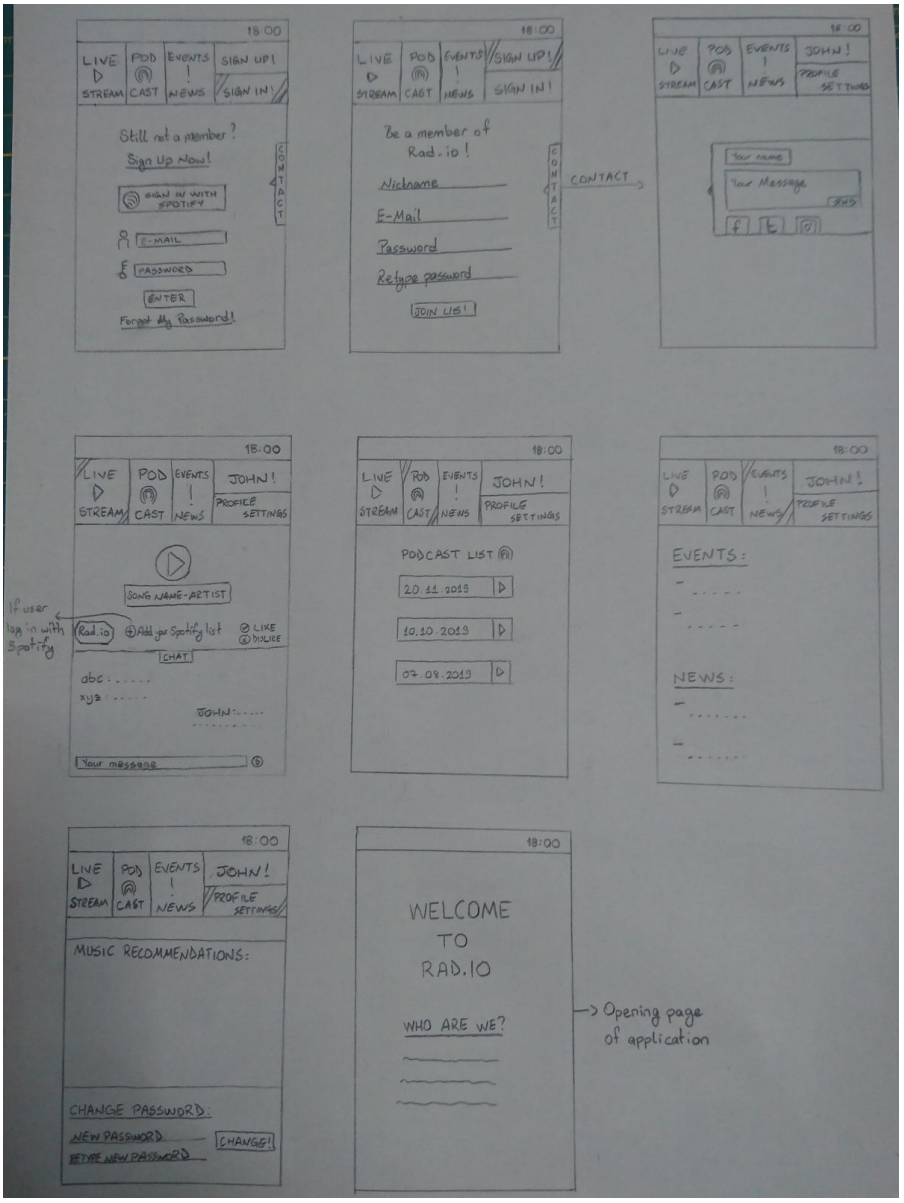


Figure 3 Rad.io UI Sketches

5. Software Design Description

5.1 Introduction

This document provides detailed information about the requirements of the radio system software. It will include the vision statement and domain analysis regarding the project. As well as designs and detailed description of the system for both mobile and web platforms.

5.1.1 Purpose

We aimed to develop a fast, interactive, and user-friendly online radio application that works on both web and mobile platforms. Our platform will use modern solutions and functionalities such as analyzing the audience and improving our user experience based on it. Minimalist and responsive design to improve usability. Scalable server infrastructure to handle the incoming traffic using modern solutions.

5.1.2 Scope and Design Goals

Nowadays, radio becomes less popular because of some applications like Spotify. Users can reach any song at anytime and they listen to people instead of radio streamers from applications like Youtube and Twitch. We think radio concept needs innovation to keep up with the times. For this reason, we aimed to develop a fast, interactive, user-friendly and innovative online radio application for both web and mobile platforms which allows users and listeners more interactivity.

The purpose of Rad.io project is to design a “Online Radio Application for Mobile and Web Platforms“ which gives users more socialization chance with different additions. Our platform will use modern solutions and functionalities such as analyzing the audience and improving our user experience based on it. Minimalist and responsive design to improve usability. Scalable server infrastructure to handle the incoming traffic using modern solutions.

There are new points of view for a radio application. We aimed to combine radio concept with new ideas which are suitable for the internet age. The Rad.io system provides people a basic and functional online radio broadcasting station with a convenient UI and new things to learn and fun timings with others.

The project is designed to be convenient, secure, helpful, and provides functionalities.

Convenient: UI of the project will be as simple as possible. Users will use this application easily to find their intended web resource as well as it minimizes the amount of tabs and pages of the user.

Security: application will have captcha system which is useful to prevent against login brute-force attacks with that even if the user has a simple password we are preventing attacker to find that out easily.

Helpful: the application will be provided a helping service through email for users who have a report or complain about an action and are free to ask and send an email whenever they want.

Functionality: application will include everything needed for the radio application. Users have their freedom to become a member of the application as they get new features provided by the system. The application will include live radio streaming service, live chat, joining surveys, listening to missed live streams (podcasts) and listening to music through Spotify integration.

5.1.3 Glossary

Table 1 Glossary of SDD

Term	Definition
UI	User Interface
Convenient	Easy to use

Brute-force attack	A brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly
Tab	Marker used to select additional web page(s) that has been opened in the browser window
API	Set of functions used by applications to communicate between other services
Module	Is a software component or part of a program that contains one or more routines
Hashes	A salt is random data that is used as an additional input to a one-way function that "hashes" data, a password, or passphrase
Services	Is a software distribution model in which a third-party provider hosts application and makes them available to customers over the internet
Subsystems	A self-contained system within a larger system
U-users	Unregistered users
R-users	Registered users
S-users	Registered users with Spotify accounts

5.1.4 References

[1] https://m204wiki.rocketsoftware.com/index.php/Application_Subsystem_development#Subsystem_procedures

5.1.5 Overview of document

This section provides information about the contents of the rest of the document as follows: Section 2, 3, 4 , and 5 describes the subsystems and services used for this project as well as

the details of the design of the system. Section 6 and 7 displays, explains and shows the design of the user interface as well as UML diagrams of the system.

5.2 Architecture design

5.2.1 Hardware/software mapping

We are proposing server client application our application will run on server.

5.2.2 Persistent data management

In our project, for persistent data management we will use PostgreSQL which is relational database management system.

5.2.3 Access control and security

We are using builtin Django authentication middleware for user authentication and authenticate decorator. We are storing passwords on database in hashed format and we are enforcing user to use difficult passwords and we are using csrf tokens in required pages to prevent from csrf attacks.

5.2.4 Global software control

Our software controls handled by used web-framework Django.

5.2.5 Boundary conditions

Our system is working 7/24 we are proposing client-server application.

5.3 Subsystem Services

5.3.1 Subsystem Definition:

The characteristics and components of a subsystem are defined to Model 204 by the system manager during a process called subsystem definition. The defined options and components are stored in the system file CCASYS. Once a subsystem has been defined, all Dictionary users can display the options and components through Dictionary.[1]

5.3.2 Subsystem Design Components:

During subsystem definition, the components listed below can be defined. These components impact various aspects of subsystem design. The following table summarizes the required component designations.

Component	Subsystem design requires designation of...
Command line global variable	(Optional) parameter global variable. The parameter global variable allows any parameters specified by a user during a subsystem login to be stored in this variable and retained when control is transferred to another subsystem.
Communication global variable and exit value	Communication global variable and exit value. The communication global variable is used to transfer control from one procedure to another. The exit value is used to leave the subsystem. Optionally, a reserved global variable is available for transferring control between subsystems.
Error global variable	Error global variable. If an error occurs while the subsystem is executing, a three-character error code is stored in this variable. This code can then be used by an error procedure to determine the action to be taken by the subsystem.
Prefix designations	Two prefixes for procedure names. These prefixes allow Model 204 to determine whether a procedure can be saved in its compiled form for later evaluation.
Processing components	Specific procedures for types of special processing. These procedures allow Model 204 to determine the flow of control within a subsystem.

5.4 Subsystem Procedures

5.4.1 Types of subsystem procedures

Subsystem development involves writing the collection of procedures that make up a subsystem. Subsystem procedures can be categorized as one of four types:

Procedure category	Performs...
Initialization	Specified operations each time the subsystem is initialized.
Login	As the entry point for each user of the subsystem.
Main processing	Specific tasks of the subsystem.
Error	Error handling.

5.4.2 Initialization Procedure

The initialization procedure is optional.If a subsystem uses an initialization procedure, the procedure name must be specified in the subsystem definition.The initialization procedure stores instructions for tasks you need to perform each time the subsystem is initialized. An example of such a task is the initialization of a particular work file[1].

5.4.3 Login Procedure

The login procedure performs the start up for each user of an application.The login procedure name must be specified in the subsystem definition.Typically, the login procedure is used to store current server table sizes in the global variable table for later reference, issue UTABLE commands to set compiler table sizes for the subsystem, and set the communication global variable to the name of the procedure that displays an initial menu[1].

5.4.4 Main Processing Procedure

Main processing procedures perform the specific tasks of the subsystem. There can be as few or as many main processing procedures as necessary for the subsystem to perform its tasks. Main processing procedures are not specified in the subsystem definition. However, each procedure must follow the procedure naming conventions and subsystem coding rules[1].

5.5 Main Subsystem Includes:

- User Management Subsystem
- Admin Management Subsystem

5.5.1 User Management Subsystem

This subsystem provides the facilities that covers all the user management functionality.Main use case that comes under this subsystem includes:

- User Registration
- User Login(With Spotify or normal log in)
- Chatting With Users(Only for registered users)
- Listening to Broadcasts(For registered users or unregistered users)
- Adding Songs to the Spotify playlist that you "like"

5.5.2 Admin Management Subsystem

This subsystem provides the facilities that covers all the admin management functionality. Main use case that comes under this subsystem includes:

- Admin Log in
- Start/Stop Broadcasts
- Change Music
- Ban User

- Create/Delete Announcement
- Create Survey

5.6 UML Diagrams

5.6.1 Use-case Diagram

Actor(s) User: The user is a person who download the application and uses it. For using the application, user must accept the Cookies once to allow identify users and save site login information. The user can register to the website to become a member of Rad.io to gain more features. The user must accept the terms and conditions and must verify their account through email.

Stakeholder(s) Project Advisor: The project advisor is responsible for delivering the project on time and on a given budget. They usually work together and guide the team that develops the project so that the goals in the project can be fulfilled correctly.

The Project Manager's goals are; To follow the project and check if it is done as requested. To ensure that the risks and problems are properly handled. Working with the group to motivate them and give the necessary support.

Project Development Team: It is a team of people who design the project given by the project advisor according to the desired characteristics and who play an active role in the project by working together within the project.

The Project Development Team's goals are; To make the project timely and correctly. To be able to complete the project in accordance with the quality and rules as much as possible. [3]

User: The user is the person who will use the project after the project is finished. User is the person who gives an idea to the project developer about the needs of the project and be reference in the project development process. The User’s Goals are;

- Accessing the desired radio station in a fast way.
- Self entertaining by chatting with other members/radio host.
- Accessing spotify songs and sharing it among other members.

Use Case Diagram Figure 1 presents a use case diagram for the Rad.io application. The system shows the operations that end users can perform.

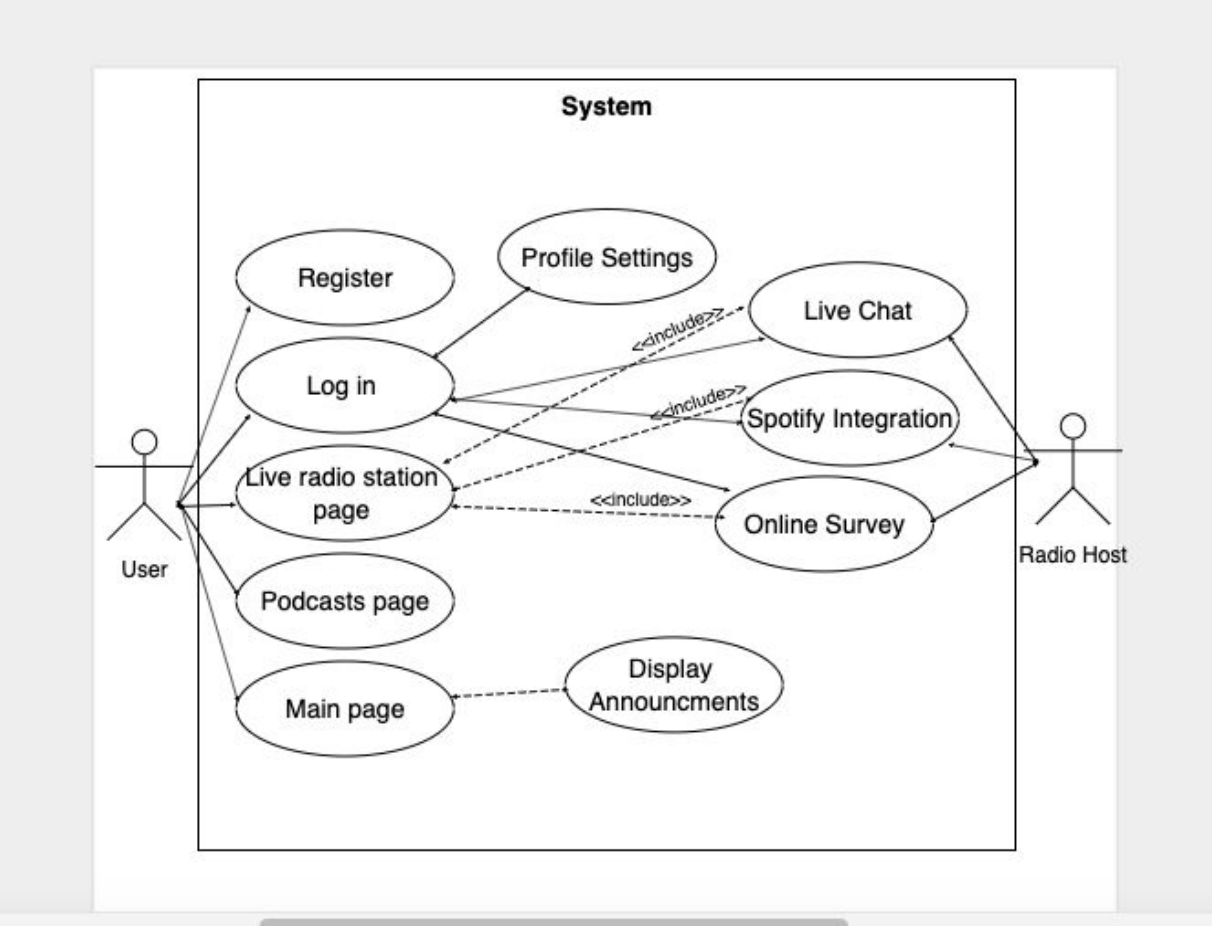


Figure 1 System Use case Diagram

Brief Description of Use Case Diagram

The use-case diagram shows us the web pages that the user and radio host can access. The user can register to the website to gain more features, non-members can also access the website but cannot have the extra features. As shown, when the user accesses the "Live Radio station page" he/she can see the features but cannot use them unless they are registered to the

system. Non-members can read the live chat messages but won't be able to chat with other members, they can see surveys but cannot participate, and so on. The radio host has the opportunity to take control of actions and operations over the live chat, spotify integration, and online survey during live streaming of the station.

5.6.2 Class Diagram

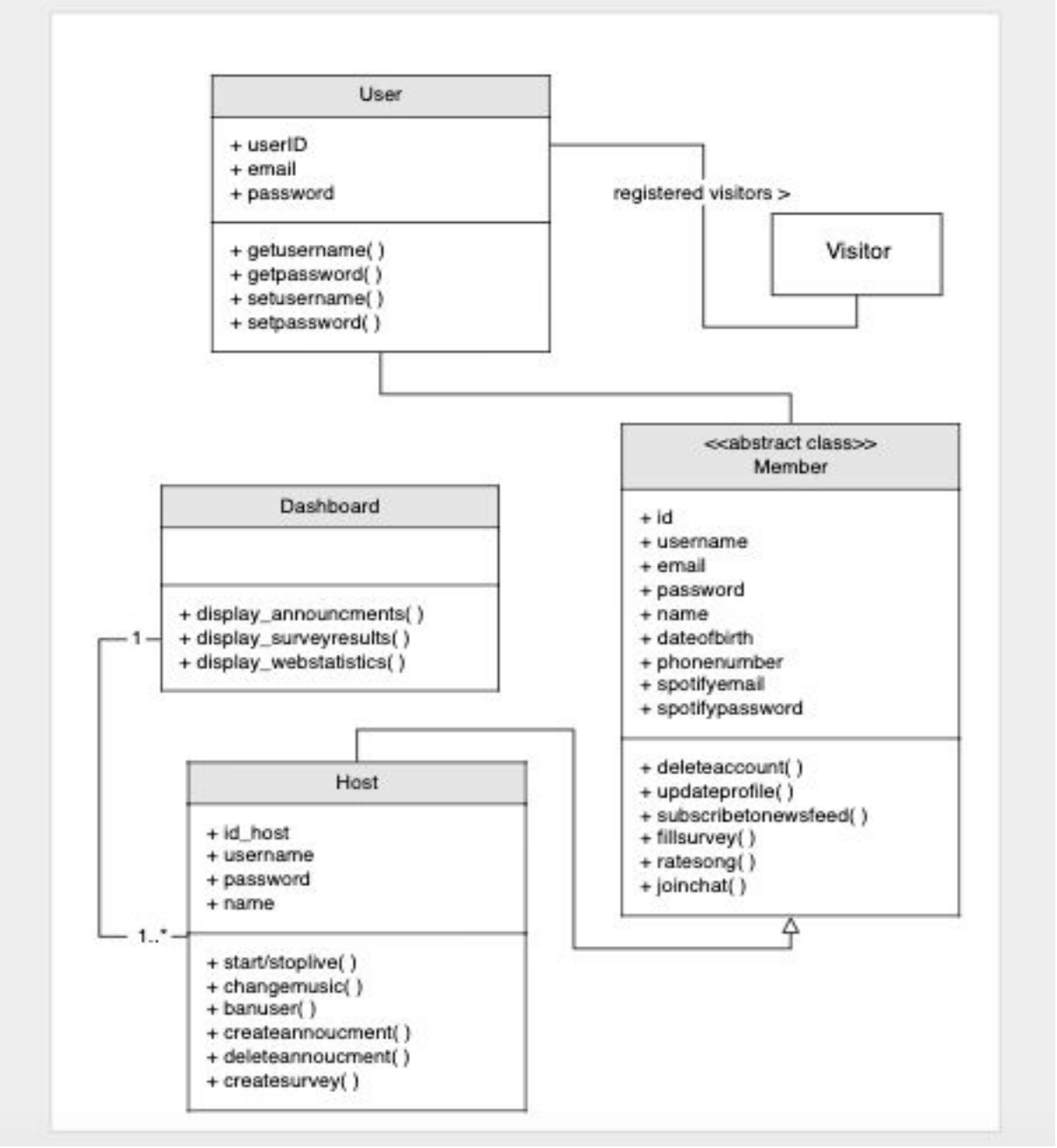


Figure 2 System Class Diagram

Brief Description of Class Diagram

The class diagram shows us the functions that the member and radio host can use. The user can register to the website to gain more features, non-members (visitors) can also access the

website but cannot have the extra features. As shown above, members who are registered to the system gain more features as shown under the "Member" entity, these functions below represent the features and actions that can be done by the entity. As we see, the "Host" entity which is the subclass, inherits the "Member" entity since you need to be a member to be a host as well but instead, in special cases, the "Host" entity acts as an admin for the system which gives him control over other users, and includes special functions in order to manage or create operations.

5.7 Interface design

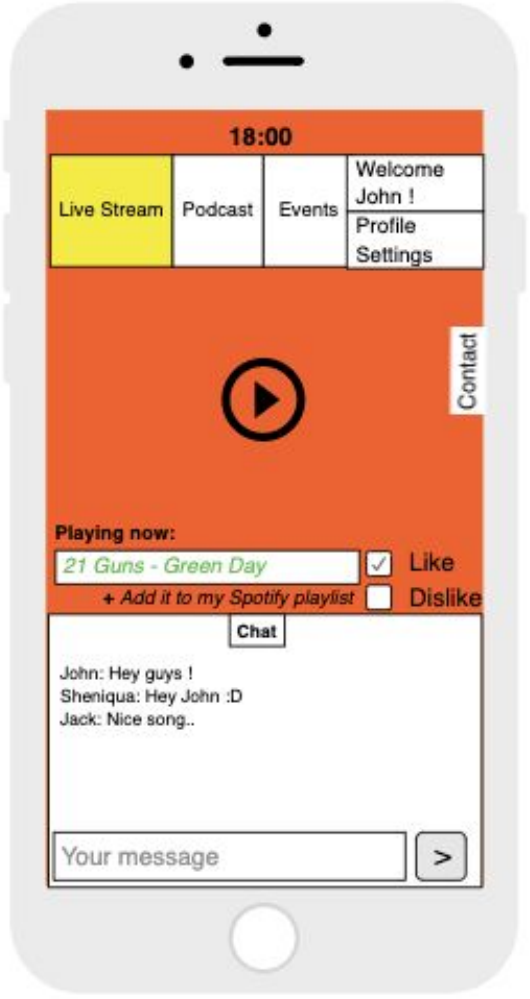
5.7.1 Overview of User Interface

In this section of the document, the layout of the pages are presented in below figures. From now on unregistered users will be referred as **U-user**, registered user will be referred as **R-user**, users which registered with Spotify will be referred as **S-user** and '**users**' word will be used for all type of users.

5.7.2 Opening Page

- The opening page has the information about the group.

5.7.3 Live Stream Page

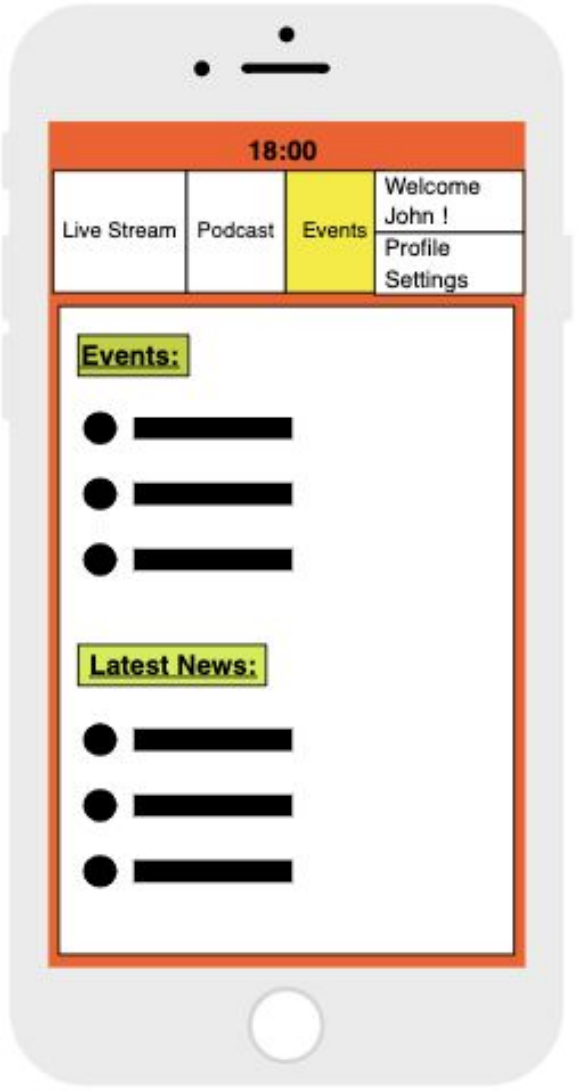


- The Live Stream Page is the interface that the users will play the song with play button. All users can write a report, a statement, or ask for information by clicking on the contact button. R-users and S-users can like or dislike options for music recommendations. Also S-users can add the song to their playlist on Spotify. There is a live chat on the main page which is active at the on air time but U-users can not write on chat, they just can read the conversation.

5.7.4 Podcasts Page

- The Podcasts Page includes records of old streams or special conversations about different topics. Users can listen with play buttons.

5.7.5 Events Page

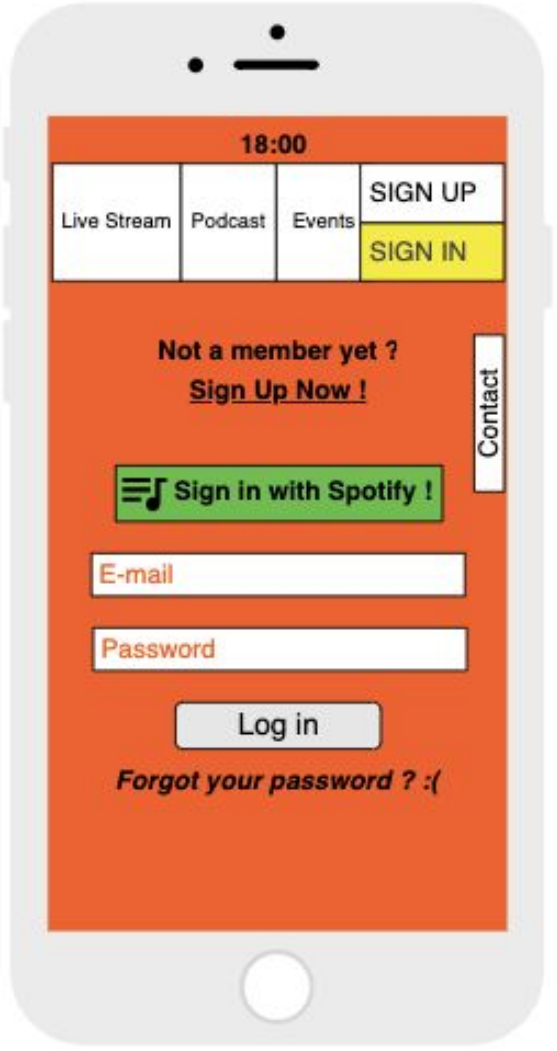


- The Events Page shows the events or news about the weekly stream program. Users can read every new improvement from this page.

5.7.6 Sign Up Page

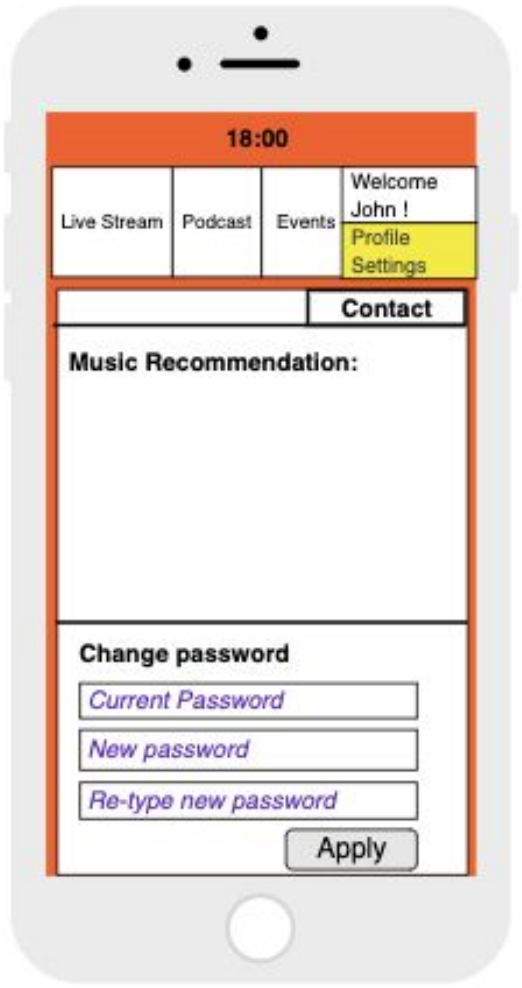
- The Sign Up Page is for registration. New users can sign up from this page with their information. After a user logged in, this button will be inactive and show Profile Settings button.

5.7.7 Sign In Page



- The Sign In Page includes two different types of log in. First type is ‘Sign in with Spotify’. With this type, users can add the songs to their Spotify playlists. With second type they do not have this feature. Also, in this page there is a link to Sign Up Page and a button for users which forgets his/her password. After a user logged in, this button will be inactive and show his/her nickname.

5.7.8 Profile Settings Page



- Just S-users and R-users can access this page. It has music recommendations for S-users and R-users. Also they can change their passwords from this page.

6. Conclusion

In this CENG407 project, we talked about the requirements specification needed for both mobile and web platforms and features for hardware and software, as well as performance requirements for the radio application. We included definitions and feature designs for the application. The proposed radio application works on both web and mobile platforms which will be established by a dedicated API on server side which allows mobile platform to use same functionalities, databases in web application. Both web and mobile platform offers user to registration and login system to allow user to use more functionalities that are required authentication. Also, we took security measures such as captcha in login, register pages, rate-limiting in input submissions, csrf protection in form submissions, general input validation and more. We got three types of user model administrative user, streamer, and standard user but we allow some functionalities to be performed without user authentication such as listening to livestream. After user login we collected data with permission by asking users to fill up some questions, and passive data collection based on user's network data for improving quality of platform and increase user experience. Also, platform has an online chat that can allow to build a community around stream. According collected data, current streamer can decide what to do next, or can give targeted advertisement to audience. In administrative pages streamer can perform given operations easily with user-friendly UI. We are developing modular system this allows us to integrate microservices around base platform and allow developers to easily add new modules and functionalities. This provides people a basic and functional online radio broadcasting station with a convenient UI and new things to learn and fun timings with others.

7. References

[1] https://en.m.wikipedia.org/wiki/Internet_radio

[2] https://en.wikipedia.org/wiki/Broadcast_delay

[3] Banta T. Vxchange. Retrieved September 27, 2019, from https://www.vxchnge.com/blog/common-causes-of-server-downtime

[4] https://en.wikipedia.org/wiki/Radio

[5] https://www.definitions.net/definition/Radio

[6] https://www.rnib.org.uk/practical-help/technology/resource-hub/guides-entertainment/online-radio

[7] https://www.pythonforbeginners.com/learn-python/what-is-django/