International University of Sarajevo
Faculty of Engineering and Natural Sciences
Software Engineering

KS Prevoz – Public transport web application

Final submission

CS308 Software Engineering

Spring 2021

Team members:

1. Harun Tucaković
2. Muhammed Musanović
3. Šejla Burnić
4. Fejsal Perva

Instructor:

Prof. Kanita Karađuzović – Hadžiabdić, PhD

Sarajevo, May 2021

Contents

1 SRS Document

Abstract

For citizens of Canton Sarajevo who use the public transport daily, the KSPrevoz is a web application that will provide arrival time of public transport vehicles in the Canton Sarajevo and appropriate routes from point A to point B. It will also make schedule updating and notification sharing easier for administration of public transport company. Unlike the current system our product will give users access to crucial information via a web application instantly and accurately. The purpose of this document is to provide an overview of our software product, its' parameters, and goals. In this document we describe the user interface, hardware, and software requirements of our system. This document is intended for the developers' team that is going to create the KSPrevoz system and application.

Document Revision History

Rev1.0 March 20, 2021 – initial version

Rev2.0 May 17, 2021 – updated use case diagram, added introduction text for each new section, put references to figures and tables in text

All functional requirements stated in the SRS have been successfully implemented.

1.1 Introduction

We have found that the current public transport information sharing system is unreliable, the schedules are faulty and not updated frequently, users often cannot find information for lines they are interested in. Also, users are not informed about changes in schedules. This application is going to be developed with the intention of improving quality and user satisfaction with public transport. Unlike the current system, our product will give users access to relevant information via web application instantly and accurately. Users of our application will be able to check public transport schedule, lines, and get routes from point A to point B. They will be able to create an account and create a personalized scheduled by making their favorites line list. The administration of the public transport company will be able to upload and edit schedules, as well as send email notifications to users. Our project fits in with business objectives of public transport companies. We expect that our application will drive the number of users of public transport up by 10% by 2022, as well as improve overall satisfaction with public transport. Expected user classes for our application are passengers (with or without account) and admins. Additionally, we assume that user satisfaction with public transport will improve by having all schedules and related information available to them on their phone and desktop. Our application depends on Google Maps API.

1.2 System Features and Use Cases

System features are a product's intended capabilities and functionalities that it will offer to users while use cases are detailed explanations of how an actor will accomplish a goal and all the necessary steps to accomplish that goal.

1.2.1 System Features

The list of all system features for KSPrevoz as well as their brief explanations, priority, user requirements and corresponding functional requirements can be seen below. In table 1 are listed all use cases and their primary actors.

FTR1 **Account:** Users can create, manage, and use personal account. Users of this application will be able to use their account to add lines to their favorites list for personalized schedule access**.**

Priority: Must have

UR1.1 Create account

FR1.1.1 The user shall be able to enter their name, email, and password to create an account.

FR1.1.2 The user shall be able to create an account using their Google account.

FR1.1.3 The system shall not allow a user to create an account with an email that is being used by another account.

UR1.2 Log into account

FR1.2.1 The user shall be able to access their account by entering their email and corresponding password on the log in page.

FR1.2.2 The user shall be able to access their account using their Google account log in.

UR1.3 Manage account

FR1.3.1 The user shall be able to change their password.

FR1.3.2 The user shall be able to change their email.

FR1.3.3 The user shall be able to delete their account.

FTR2 **Favorites list:** Users that have an account and are logged in will be able to manage their favorites list. They will be able to add and remove lines to this list to be able to create personalized view of schedules.

Priority: Must have

UR2.1 Add line to the list

FR2.1.1 The user shall be able to add a line to the list by clicking “Add to favorites” on the list of all lines and on the page with opened schedule for that specific line.

UR2.2 Remove line from the list

FR2.2.1 The user shall be able to remove line from the favorites list by clicking “Remove” on the list of all favorite lists

UR2.3 View their favorites list

FR2.3.1 The user shall be able to view favorites list immediately after they logged in and they should have option to open up favorites list from the navigation menu.

FTR3 **Schedule:** Viewing and modifying transport schedules. Admin account will have option to add new, update and remove existing schedules/lines. User account will be able to view all the schedules.

Priority: Must have

UR3.1 Modify schedule

FR3.1.1 The admin shall be able to add a new line to the schedule.

FR3.1.2 The admin shall be able to update schedule of any line.

FR3.1.3 The admin shall be able to remove a line from the schedule.

UR3.2 View schedule

FR3.2.1 The user shall be able to search a line by the name of the line.

FR3.2.2 The system shall be able to display schedule for a line as a table.

FTR4 **Information sharing:** Sharing of important information via notifications. Update of any schedule will send automatic email notification to all users that have that particular schedule/line in their favorites list. Admin will be able to send custom email notifications to users.

Priority: Should have

UR4.1 Send notification

FR4.1.1 The system shall be able to send email notifications to users when admin updates schedule.

FR4.1.2 Admin shall be able to send custom message to all users that have an account as an email notification.

UR4.2 Receive notification

FR4.2.1 The user shall be able to receive notifications about changes in the schedule via email address for the lines that are in their favorites list.

UR4.3 View notification

FR4.3.1 The system shall be able to display all notifications that admin sent as a list.

FTR5 **Path finding:** Finding optimal route between two points.

Priority: Could have

UR5.1 Find routes between two points

FR5.1.1 The user shall be able to find routes between point A and point B.

Primary Actor	Use Cases
Guest User	1. Sign up 2. Log in 3. View schedule 4. Find route
Logged in User	5. Manage account 6. Add line to favorites list 7. Remove line from favorites list 8. View favorites list 9. Receive email notifications
Admin	10. Add new line 11. Delete line 12. Modify schedule 13. Send email notifications 14. View notifications

Table 1 Use cases

1.2.2 Use Cases

As stated above use cases are detailed explanations of how an actor will accomplish a goal and all the necessary steps to accomplish that goal. A use case diagram on the other hand is a abstract graphical depiction of a system where all actors are listed and their respective use cases. For the use case diagram for KSPrevoz refer to Figure 1 below.

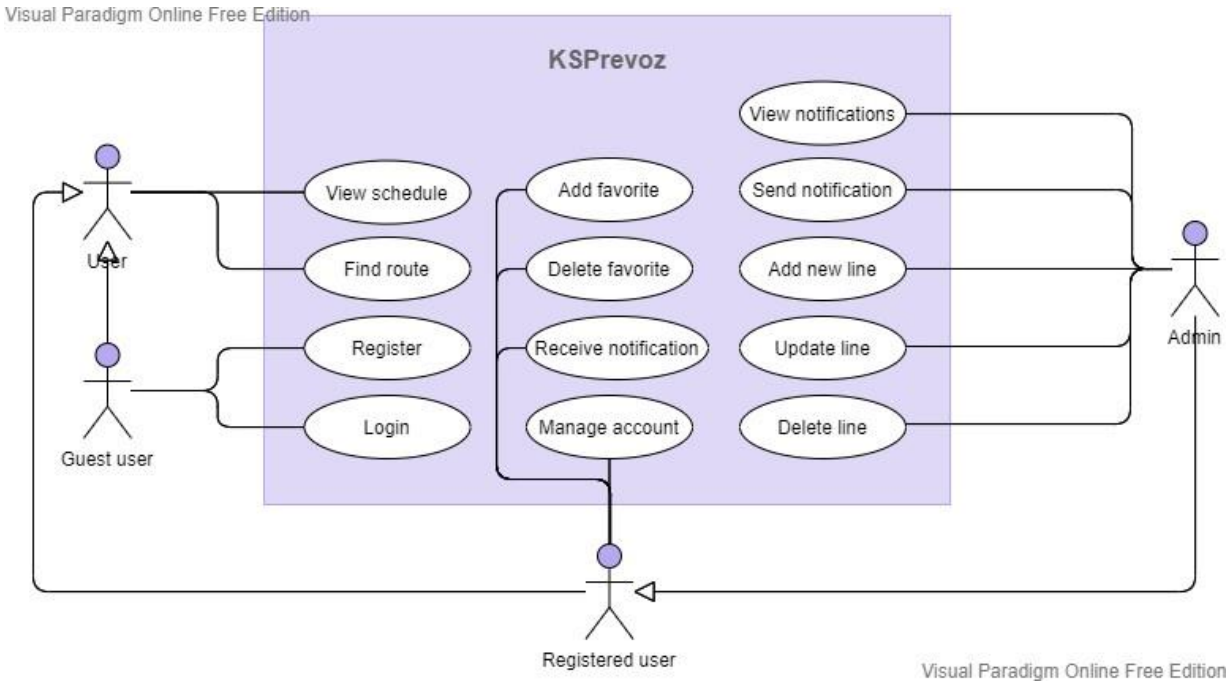

Figure 1 Use case diagram

1.2.2.1 Detailed Use Cases

The list of use cases for KSPrevoz and their detailed explanations can be read below.

Use case ID and title: UC-1 Sign up
Description: A passenger can create an account. The account allows the user to login, use favorites list, and receive notifications.
Priority: Must have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.

Postconditions: An account with passenger's information is created in the database.
Basic Flow:

1. Passenger clicks on Sign up.
2. System loads the Sign-up page.

3. Passenger fills out the necessary data needed to submit.
4. Passenger clicks on Sign up button.
5. System creates the account with the given information by adding the information to the database.
6. System redirects user to the home page.

Alternate flow:

1. Passenger clicks on Sign Up with Google Account.
2. System opens a new window to complete the sign up with a Google Account.
3. Passenger fills out the necessary data needed to log into Google Account and clicks on Log In.
   4. Passenger agrees to give access to Google Account Information.
4. System creates the account with the given information by adding the information to the database.
5. System redirects user to the home page.

What can go wrong: Passenger inputs data in wrong format in the required fields. System displays an error message prompting the user to correct the input data.
Assumptions: Admin will have their account made by the company.

Use case ID and title: UC-2 Log in
Description: The passenger logs into their account with the email address and password information they used when they were signing up.
Priority: Must have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Passenger is not logged in.

Postconditions: Passenger is logged into their account and is redirected to Home page.
Basic Flow:

1. Passenger clicks on Login.
2. System loads the Login page.
3. Passenger enters the necessary information.
4. Passenger clicks the Login button.
5. System checks the information and if it is correct, lets the user log in.
6. System redirects the passenger to the Home page.

Alternate flow:

1. Passenger clicks on Login.
2. System loads the Login page.
3. Passenger clicks on Login using Google account.
4. Passenger, if not already logged in, enters information to log into their Google account.
5. System logs in the passenger.
6. System redirects the passenger to the Home page.

What can go wrong: The passenger does not enter all required information in the input fields or inputs incorrect information. The system displays an error message and prompts the user to enter the needed or valid information.

Use case ID and title: UC-3 View schedule
Description: A passenger can see the schedule of a line.
Priority: Must have
Preconditions:

1. Server is online.
2. A passenger is connected to the internet.

Postconditions: A passenger sees the schedule for the given line.
Basic Flow:

1. Passenger clicks on “Schedules” in navigation menu.
2. System loads Schedule page.
3. Passenger selects line.
4. System displays schedule for the selected line.

Alternate flow:

3. The passenger searches for a line in the search bar.
4. Systems displays available options for that line.
5. Passenger selects line.
6. System displays schedule for the selected line.

What can go wrong: Passenger searches for a line that does not exist. The system displays error message saying that the line does not exist.

Use case ID and title: UC-4 Find route
Description: A passenger can select the route they want between two points.
Priority: Could have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Google Maps API is working.

Postconditions: A route between two points is displayed and appropriate transport options.
Basic Flow:

1. Passenger clicks on “Find route” in navigation menu.
2. System loads the page.
3. Passenger inputs starting and ending point of their route.
4. System generates and displays the route and appropriate transport options according to the information provided by the user.

Alternate flow: -
What can go wrong: -

Use case ID and title: UC-5 Manage account

Description: The passenger can manage their account which allows them to change the account information.
Priority: Must have
Trigger: The passenger clicks on Edit account on the Account page.
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Passenger is logged into their account.

Postconditions: The passenger's account information is changed.
Basic Flow:

1. Passenger clicks on Edit account on Account page.
2. System loads the Edit account page.
3. Passenger enters the necessary information that they wish to change.
4. Passenger clicks the Apply button.
5. System makes changes in the database.
6. System redirects the passenger to the Home page.

Alternate flow: -
What can go wrong: Passenger enters data in invalid format. The system displays an error message and prompts the user to enter valid data.

Use case ID and title: UC-6 Add line to favorites list
Description: A passenger will be able to add a line to their favorites list.
Priority: Must have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Passenger is logged in.

Postconditions: A line is added to the favorites list of the user.
Basic Flow:

1. Passenger clicks on "Add to favorites" on the schedule page.
2. System displays pop-up notification that line is added to favorites.

Alternate flow: -
What can go wrong: -

Use case ID and title: UC-7 Remove line from favorites list
Description: A passenger will be able to remove a line from their favorites list.
Priority: Must have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Passenger is logged in.

Postconditions: A line is removed from the favorites list of the user.
Basic Flow:

1. Passenger clicks on “Remove from favorites” on the schedule page or on the list of favorites.
2. System displays pop-up notification that line is removed from favorites.

Alternate flow: -
What can go wrong: -

Use case ID and title: UC-8 View favorites list
Description: A passenger will be able view their favorites list.
Priority: Must have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.
3. Passenger is logged in.

Postconditions: List of favorites lines is displayed on the screen.
Basic Flow:

1. Passenger clicks on “Favorites List” on the navigation menu.
2. System displays favorites list.

Alternate flow: -
What can go wrong: -

Use case ID and title: UC-9 Receive email notifications
Description: System sends passengers a notification if there are any changes to the schedule.
Priority: Should have
Preconditions:

1. Server is online.
2. Passenger is connected to the internet.

Postconditions: Passenger receives email notification with information about schedule updates and other relevant information.
Basic Flow:

1. Admin modifies schedule for a line.
2. System sends notification to passengers about the change.

Alternate flow: -
What can go wrong: Passenger has their notifications turned off and therefore the system does not send him an email.

Use case ID and title: UC-10 Modify schedule
Description: An admin can modify the schedule when it is needed. The modifications can be adding, removing, or editing lines.
Priority: Must have
Preconditions:

1. Server is online.
2. Admin is connected to the internet.
3. Admin is logged into their account.

Postconditions:

1. A line is added, removed, or edited depending on what the user wanted.
2. The request is recorded in the database.

Basic Flow:

1. Admin clicks on “Delete” or “Edit” for a certain line.
2. System displays corresponding form i.e., Remove form for removing and Editing form for editing.
3. Admin enters necessary information to complete request and clicks Apply.
4. System checks if entered data is valid and displays message that line was removed or edited.

Alternate flow:

1. Admin clicks “Add line” in Schedule tab.
2. System displays form for adding new line.
3. Admin enters necessary information for line and clicks “Create new line”.
4. System checks if entered information is valid and if all fields are filled up.
5. System creates new line, adds it to the database and displays confirmation message on the screen.

What can go wrong: Admin enters incorrect data while adding or editing a line. The system displays an error message in response and prompts the user to enter data in valid format or cancel the operation.

Use case ID and title: UC-11 Send email notifications

Description: An admin sends other users a notification. Notification can be regarding the changes in the schedule or any relevant news regarding public transport.

Priority: Should have

Preconditions:

1. Server is online.
2. Admin is connected to the internet.
3. Admin is logged into their account.

Postconditions: Email notification is sent to passengers.

Basic Flow:

1. Admin makes a modification to the schedule.
2. System modifies the schedule in the database.
3. System sends notification to the passengers who have the line saved in their favorites list.

Alternate flow:

1. Admin clicks on Send notification option in Notifications tab.
2. System loads form for creating a notification.
3. Admin fills in the required fields.
4. Admin submits the form.
5. System sends notifications to the passengers that have an account.

What can go wrong: Passenger has their notifications turned off and therefore the system does not send him an email.

Use case ID and title: UC-12 View notifications
Description: An admin views list of all sent notifications.
Priority: Should have
Preconditions:

1. Server is online.
2. Admin is connected to the internet.
3. Admin is logged into their account.

Postconditions: List of notifications sent by the admins is displayed on the screen.
Basic Flow:

1. Admin clicks on "Notifications" in navigation menu.
2. System displays list of sent notifications.

What can go wrong: -

1.2.3 Non-Functional requirements

Non-functional requirements are constrains put on the developed system as a whole. They provide additional value to the user and are critical to the success of a software product. The successful implementation of non-functional requirements ensures the user that the software will work as intended and be reliable. Non-functional requirements for KSPrevoz are listen and briefly explained below in Table 2.

Requirement	Definition	More details
NFR1.: Hardware Interface – I	Minimum hardware requirements for the server.	**Operating system:** Windows 10 **Processor:** Intel i9 8th gen **Disk space:** 500GB **RAM:** 32GB DDR4
NFR2.: Hardware Interface – II	Minimum hardware requirements for the client.	**Operating system:** Windows 7 **Processor:** Pentium 4 **Disk space:** not important **RAM:** 2GB DDR3
NFR3.: Software Interface	Software requirements for the system.	**Database:** MySQL relational database
NFR4.: Communication Interface	User needs internet connection to access the system via browser.	Internet connection

NFR5.: Availability	What margin of failure is acceptable for the system and how often the system will/will not be available for use.	**AVL-1:** The system shall be at least 95% available during the whole week between 5 A.M. and midnight Central European Time. **AVL-2:** Down time that is excluded from the calculation of availability consists of maintenance scheduled during hours from 12:00 A.M. through 4 A.M. Central European Time every day of the week.
NFR6.: Performance	How many user quarries the system can process and how responsive the system is.	**PER-1:** Response time after clicking a tab in the menu should be no more than 1.0 second. **PER-2:** The system shall be able to handle min 50 and max 150 queries in 1.0 second.
NFR7.: Usability	How easy it is to learn to use the system and how intuitive it is.	**USY-1:** 90% of users who have never used the application before should be able to find the line they are searching for within 1.5 minutes on average and maximum 3 minutes.
NFR8.: Security	Solutions and expectation of the system regarding its' security system.	**SEC-1:** Only users with admin privilege shall be able to modify schedules and handle notifications. **SEC-2:** Admins shall be able to perform admin privileged actions only within the company they are working for.
NFR9.: Design and implementation constrains	Constrains on the design and implementation of the system.	**CON-1:** Use Google Maps API for displaying locations of vehicles, stops and lines. **CON-2:** Web application shall be supported for Chrome, Firefox, Safari, Opera, Brave. **CON-3:** NoSQL databases should be avoided.
NFR10.: Internationalization and localization	Availability of the system in other languages, input support for different writing systems and measuring standards used.	**IL-1:** The system shall be available in Bosnian and English (British English). **IL-2:** The metric system for measuring distance shall be used. **IL-3:** Latin character encoding will be used.

Table 2 Non-functional requirements

1.3 Release plan

A release plan is a roadmap that reflect expectations about the time needed to implement a feature, the increment at which it is going to be created, it's priority and dependent features. The release plan for each of KSPrevoz can be seen in Table 3.

Requirement	Duration	Increment	Priority	Dependencies	Release
FTR1 – Account	4 days	1	Must have	-	1
FTR2 – Schedule	5 days	1	Must have	-	1
FTR3 – Favorites list	2 days	1	Must have	FTR1, FTR3	1

FTR4 – Information sharing	1 day	2	Should have	FTR1, FTR2	1
FTR5 – Path finding	4 days	2	Could Have	-	1

Table 3 Release plan

1.4 Timeline

A timeline usually depicts of how a system is to be built with respect to time, with time on the x-axis and features to be implemented on the y-axis. The timeline for KSPrevoz is shown in Figure 2.

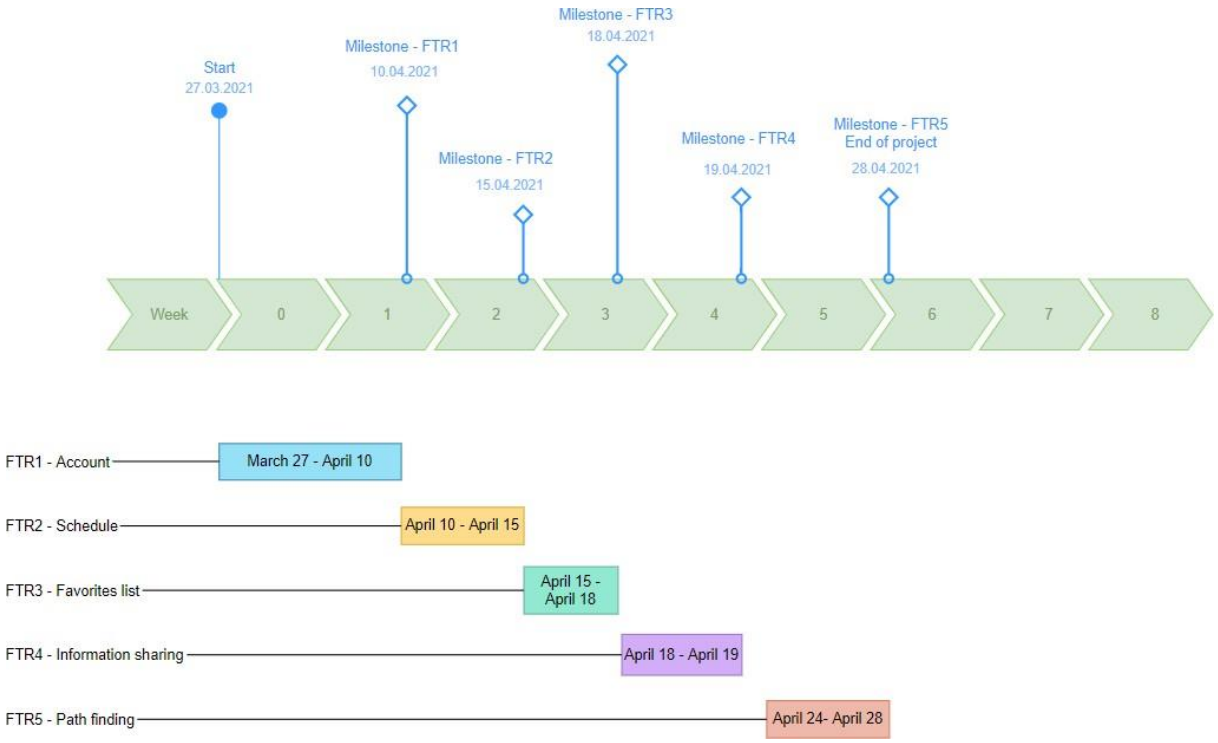

Figure 2 KSPrevoz Timeline

1.5 System evolution

We expect that after initial release all business objectives will be met by public

transport companies. KSPrevoz has potential to be expanded with more features to provide additional functionalities to its' users, e.g. electronic subscriptions, ETA option using GPS devices installed in vehicles, etc. It has the potential to also be released as a mobile application to offer more flexibility and mobility. These new and additional features, of course, depend on the budget provided for the implementation and how much they would align with the business objectives of public transport companies.

1.6 Technologies used

The KSPrevoz web application was done in the JavaScript programming language. The frontend was implemented using React.js (with Redux as state manager) and CSS while the backend was done using Node.js, Express, and mySQL for database management. Google Maps API, Google Drive API, and OAuth2 protocol were used in this project as well.

Appendix A: Glossary

Admin – company management personnel that administrates and manages the system

API – application programming interface

Passenger – person that uses public transport

Server – machine that hosts web application backend (including system database)

Contribution table

Since we did not have to make a lot of changes the whole team got in a Discord call, as usual, and together changed the SRS Document according to the feedback that we got from the professor for the first version of the SRS document.

2 Design document

Document revision history

Rev1.0 April 19th, 2021 – initial version

Rev2.0 May 17th, 2021 – added introduction text to every new section, put references to tables and figures in text

2.1 Objectives

This application is going to be developed with the intention of improving quality and user satisfaction with public transport information sharing system. Unlike the current system, our product will give users access to relevant information via web application instantly and accurately. Users of our application will be able to check public transport schedule, lines, and get routes from point A to point B. They will be able to create an account and create a personalized scheduled by making their favorites line list. The administration of the public transport company will be able to upload and edit schedules, as well as send email notifications to users. The purpose of this document is to provide an overview of our software product design, its' components, attributes, relationships among components and how they interact with each other.

2.2 System design

Regarding the class diagram, we have identified 6 classes: Line, User (and their sub-classes: RegisteredUser, GuestUser and AdminUser who is sub-class of RegisteredUser) and Notification. Line is being viewed by User and managed by AdminUser. When it comes to Notification, it is being sent by AdminUser and received by RegisteredUser. We used Visual paradigm in to draw the class diagram. The complete class diagram can be seen in figure 3.

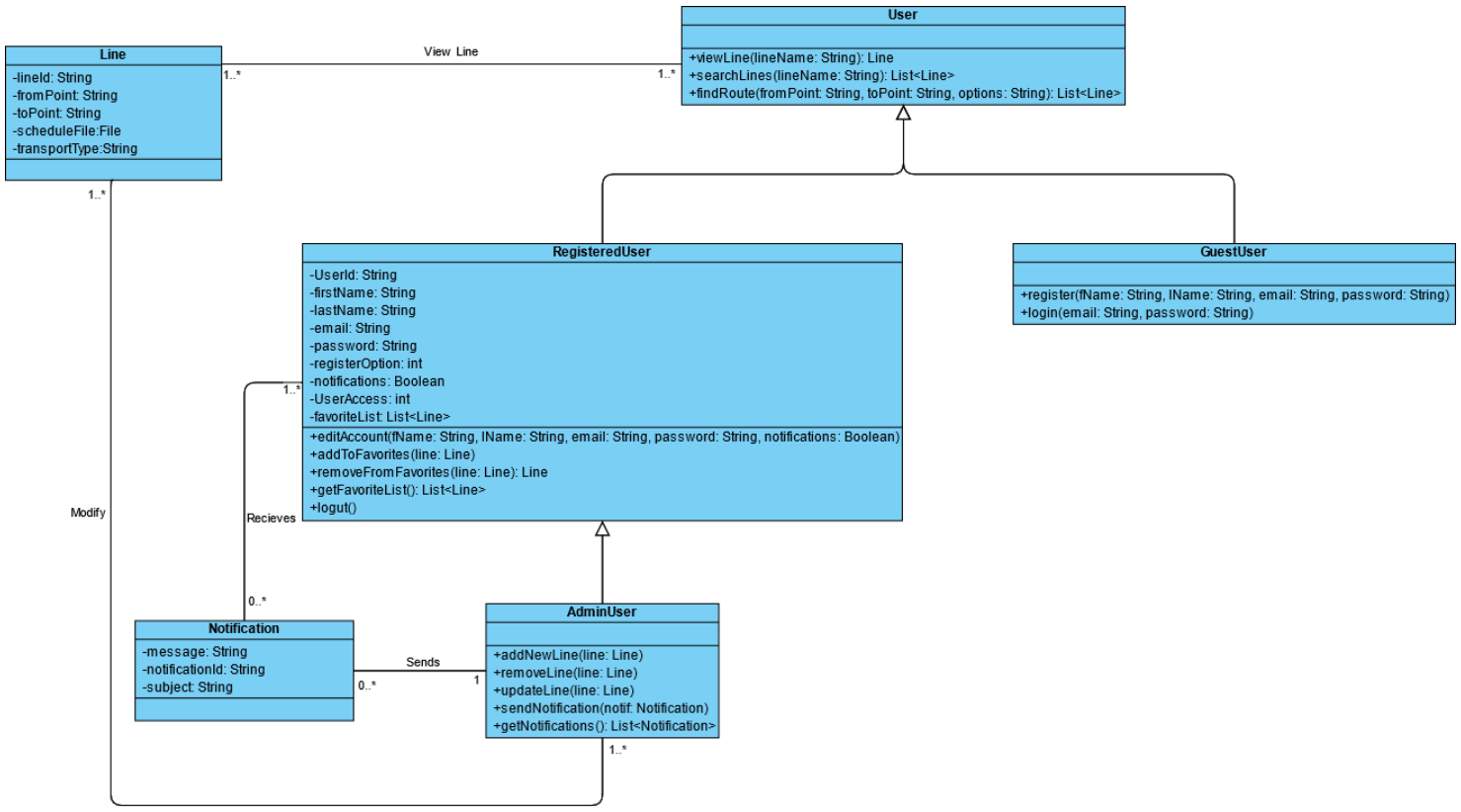

Figure 3 Class diagram

2.3 Use case diagram

When it comes to use case diagrams, we used them to depict what each user class will be able to do. For our use case diagram, we have identified four user classes: User (which is the super class of the other three classes), Guest user, Registered user (superclass of Admin), and Admin. As a result of these user classes, we have identified 13 use cases. The detailed use case diagram is shown in Figure 4.

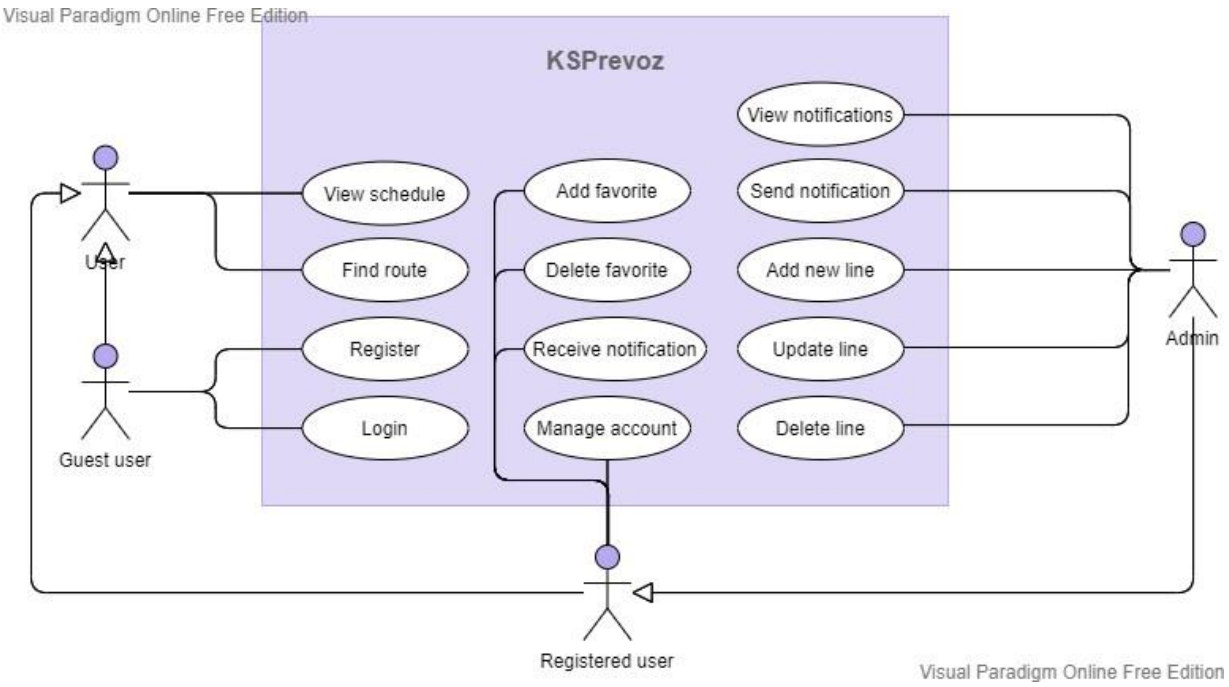

Figure 4 Use case diagram

2.4 Dynamic models

Concerning dynamic models, we have 13 sequence diagrams that depict the sequence of actor's actions and system responses that take place during realization of each use case. All the sequence diagrams are shown in figures 5-17.

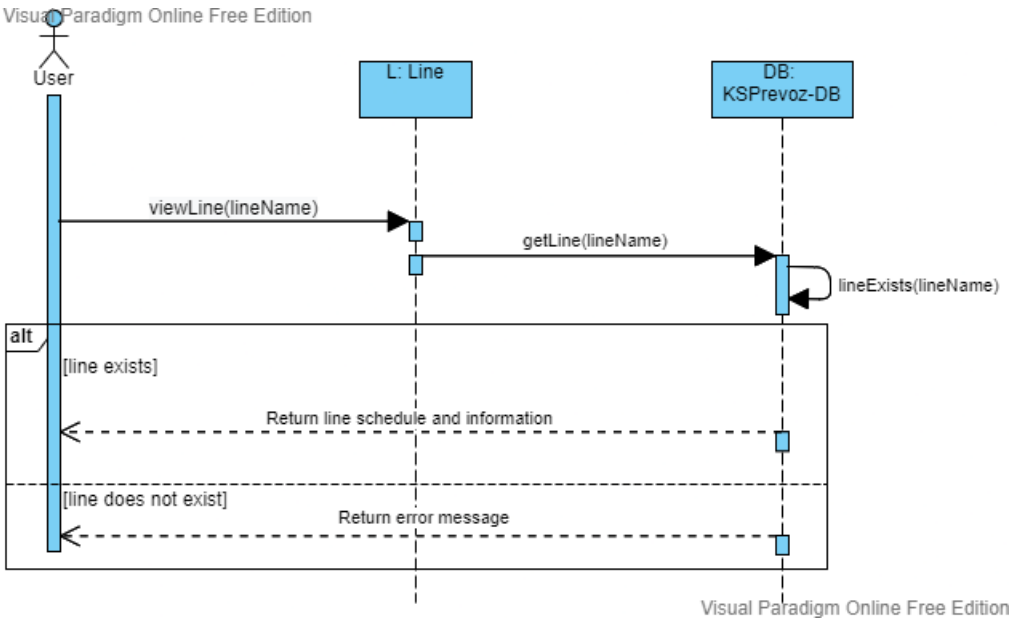

Figure 5 Sequence diagram for View Schedule

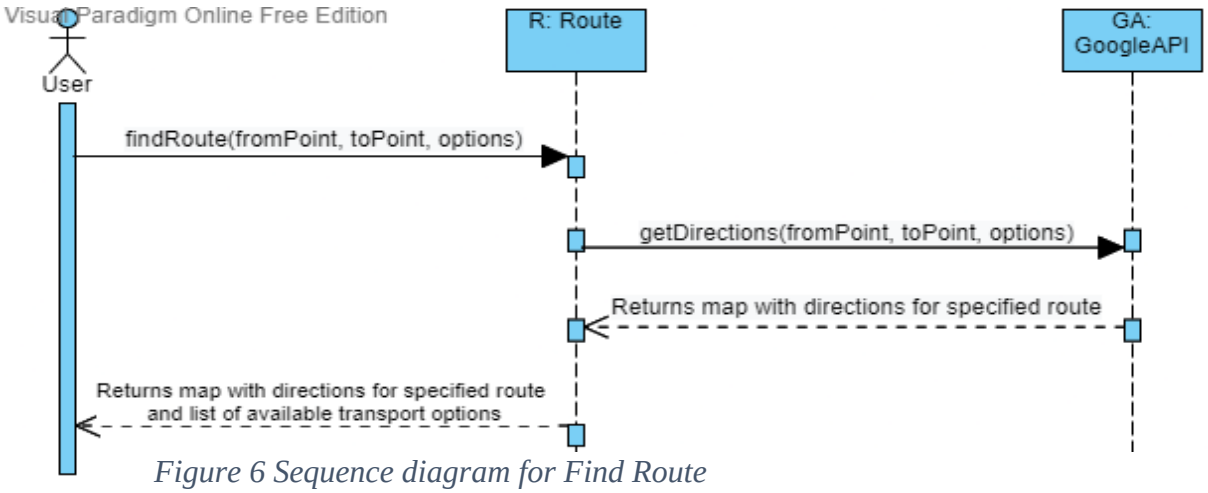

Figure 6 Sequence diagram for Find Route

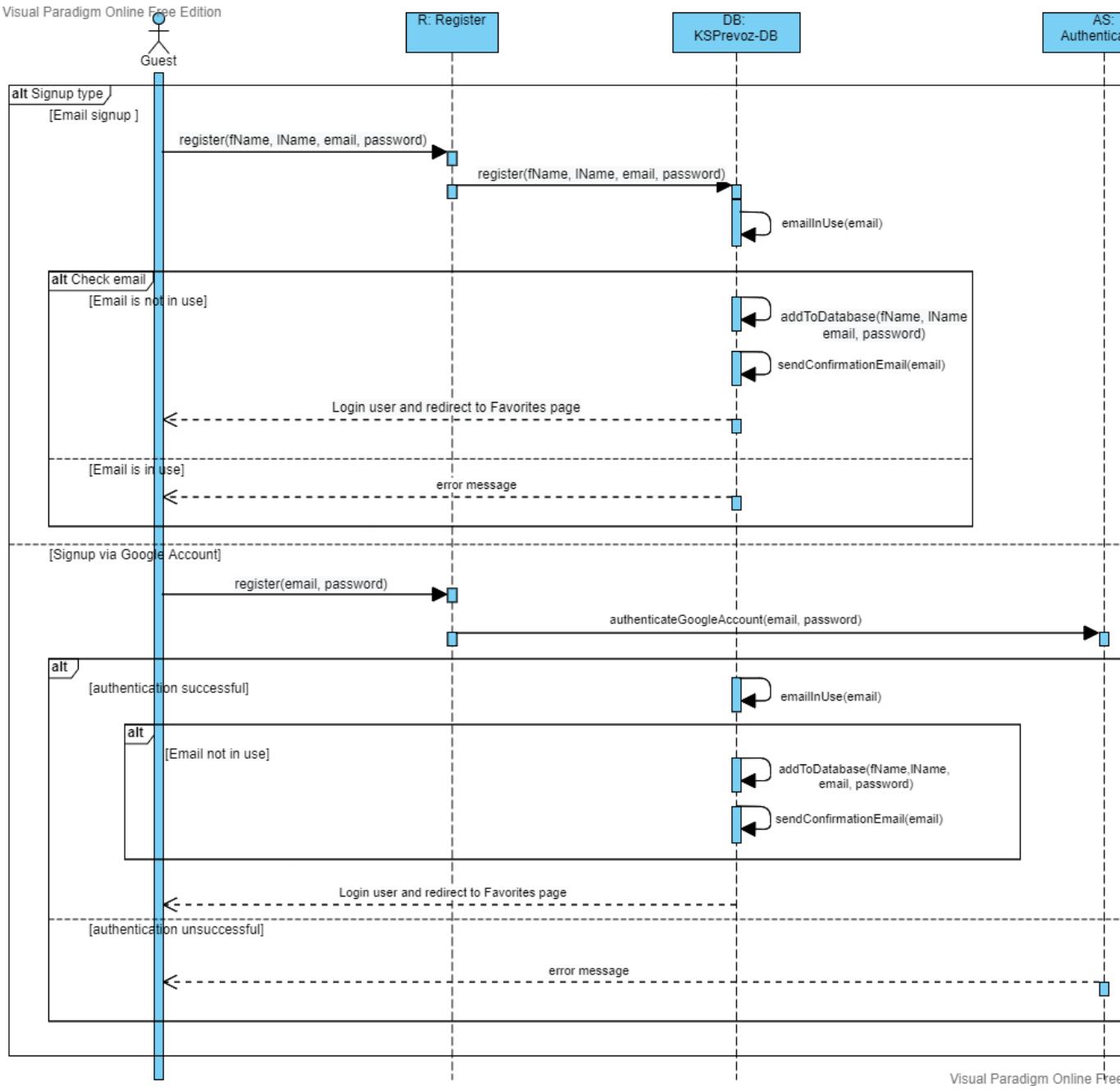

Figure 7 Sequence diagram for Register

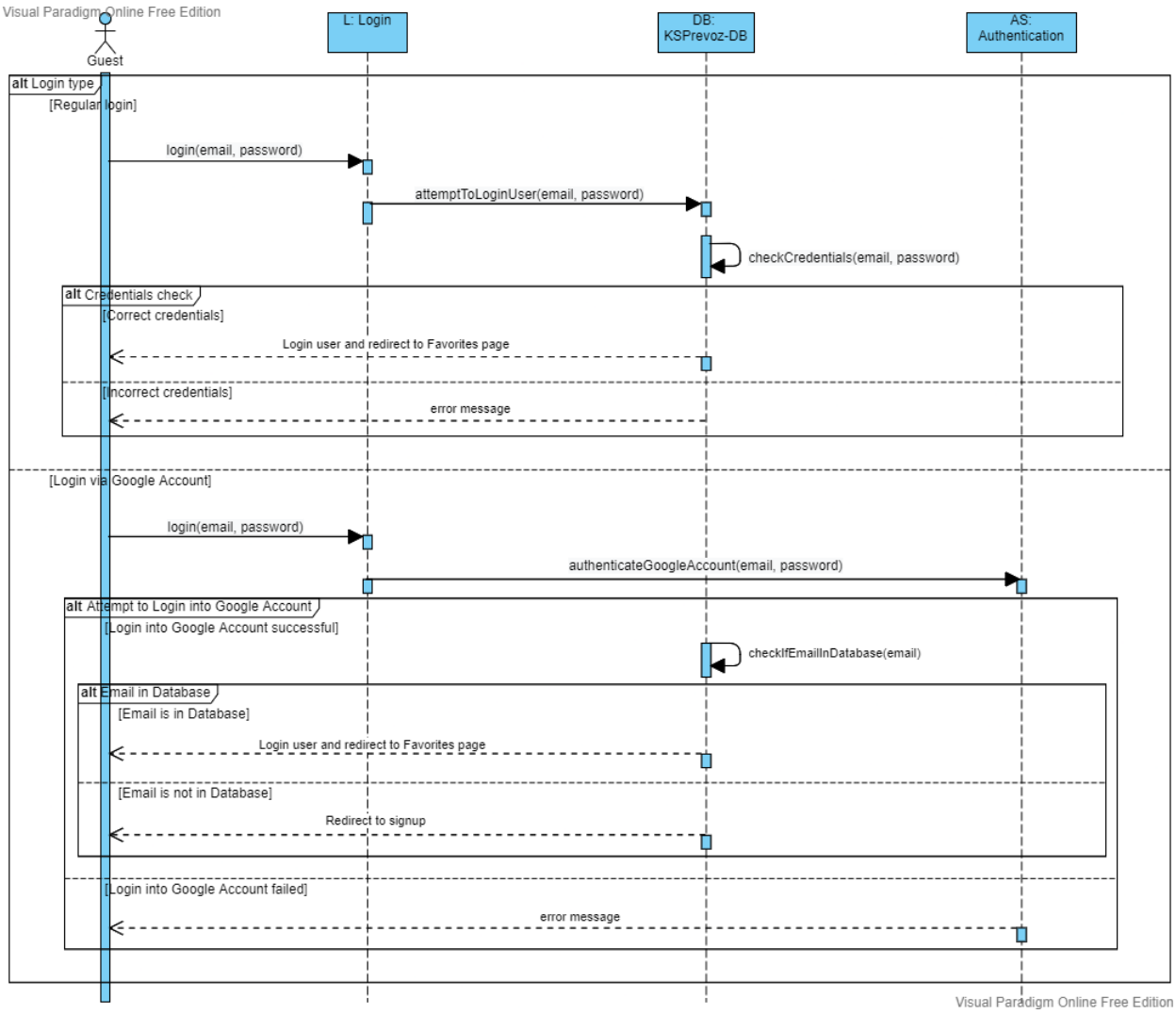

Figure 8 Sequence diagram for Login

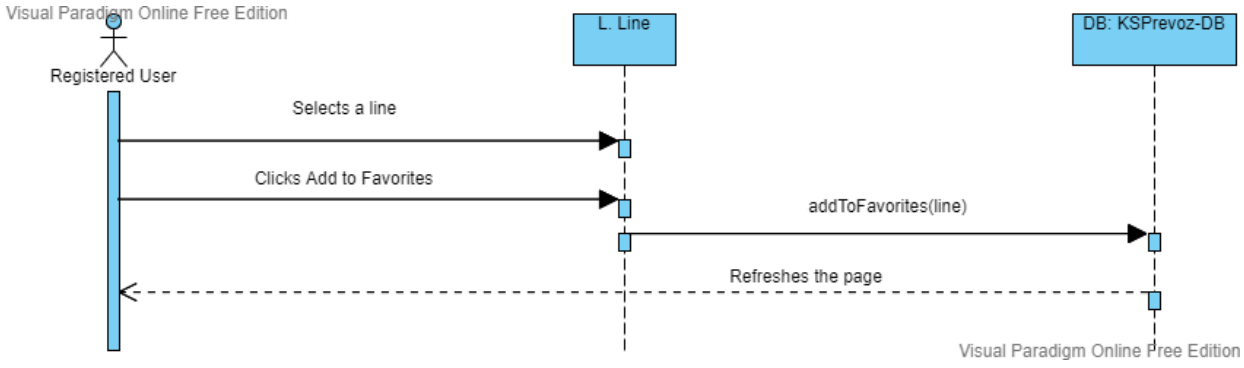

Figure 9 Sequence diagram for Add favorite

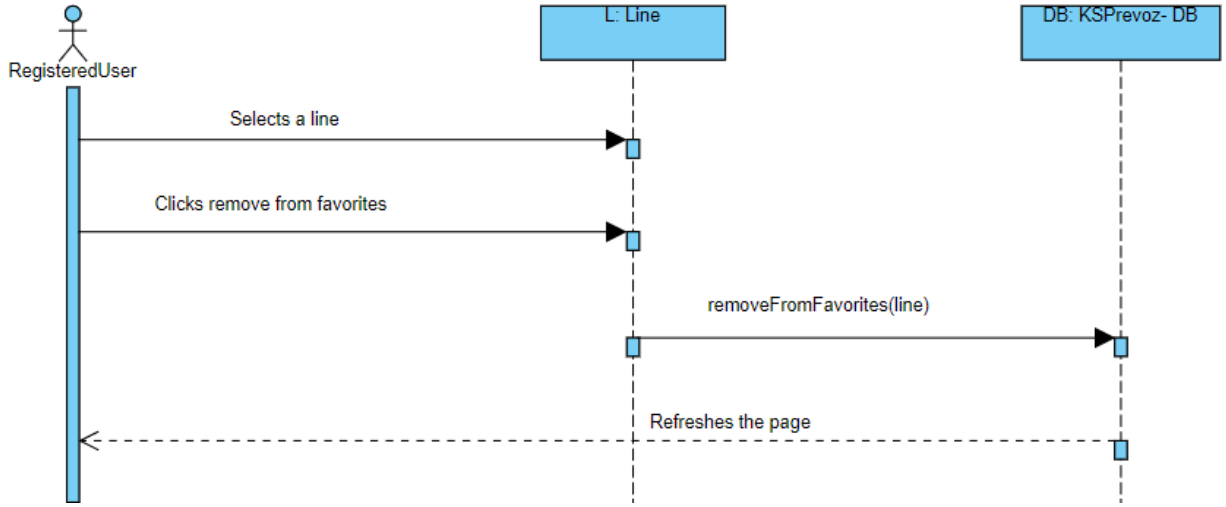

Figure 10 Sequence diagram for Remove favorite

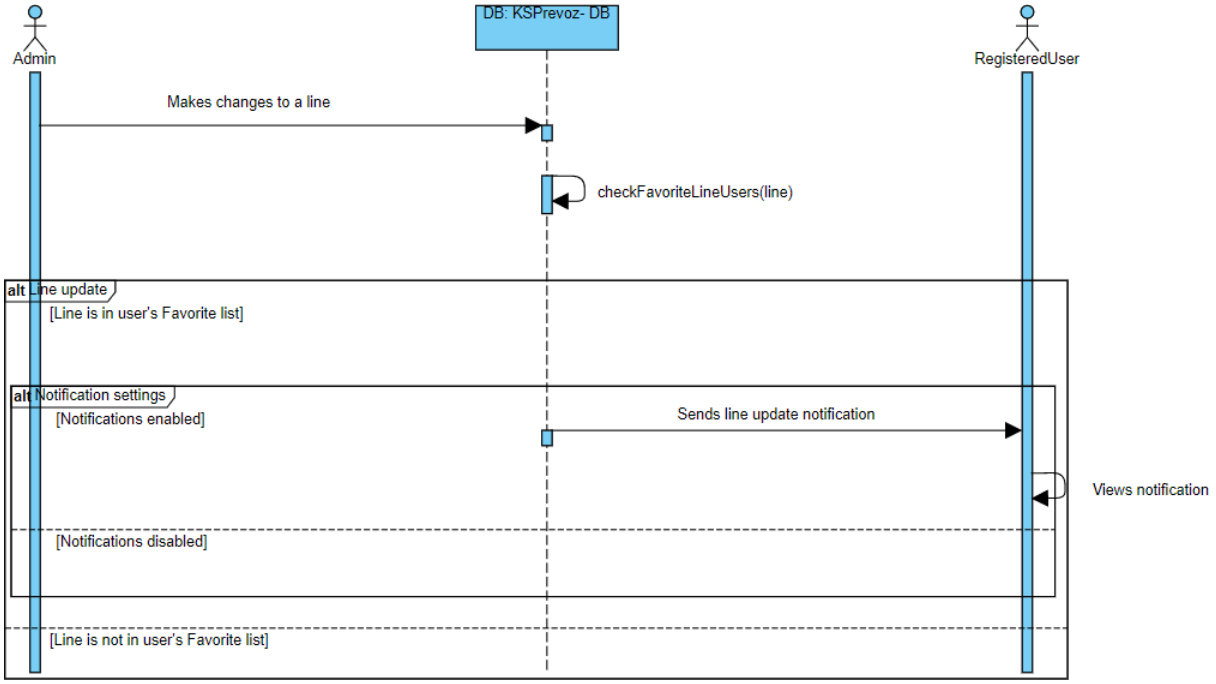

Figure 11 Sequence diagram for Receive notification

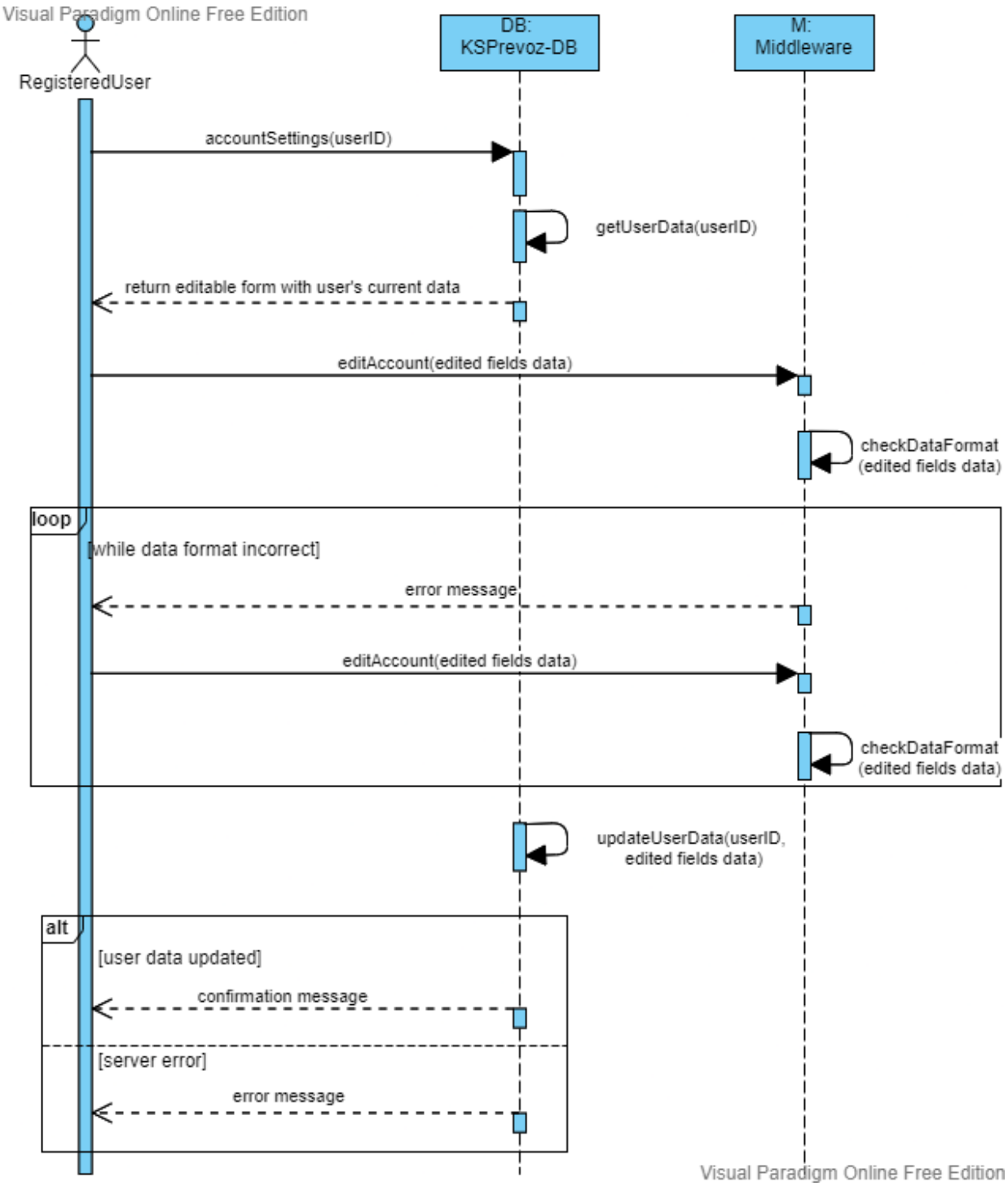

Figure 12 Sequence diagram for Manage account

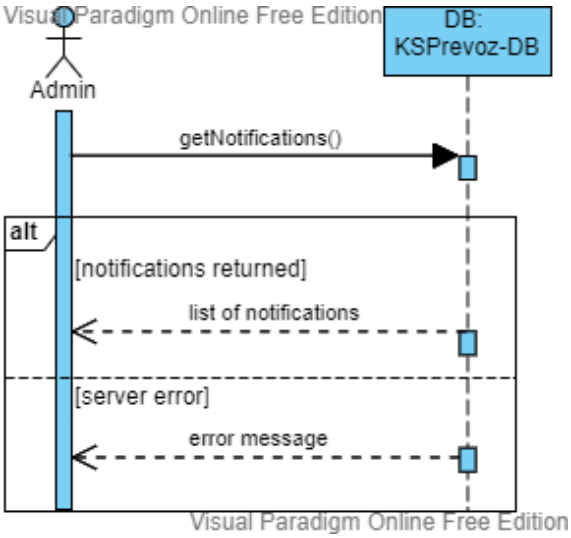

Figure 13 Sequence diagram for View notification

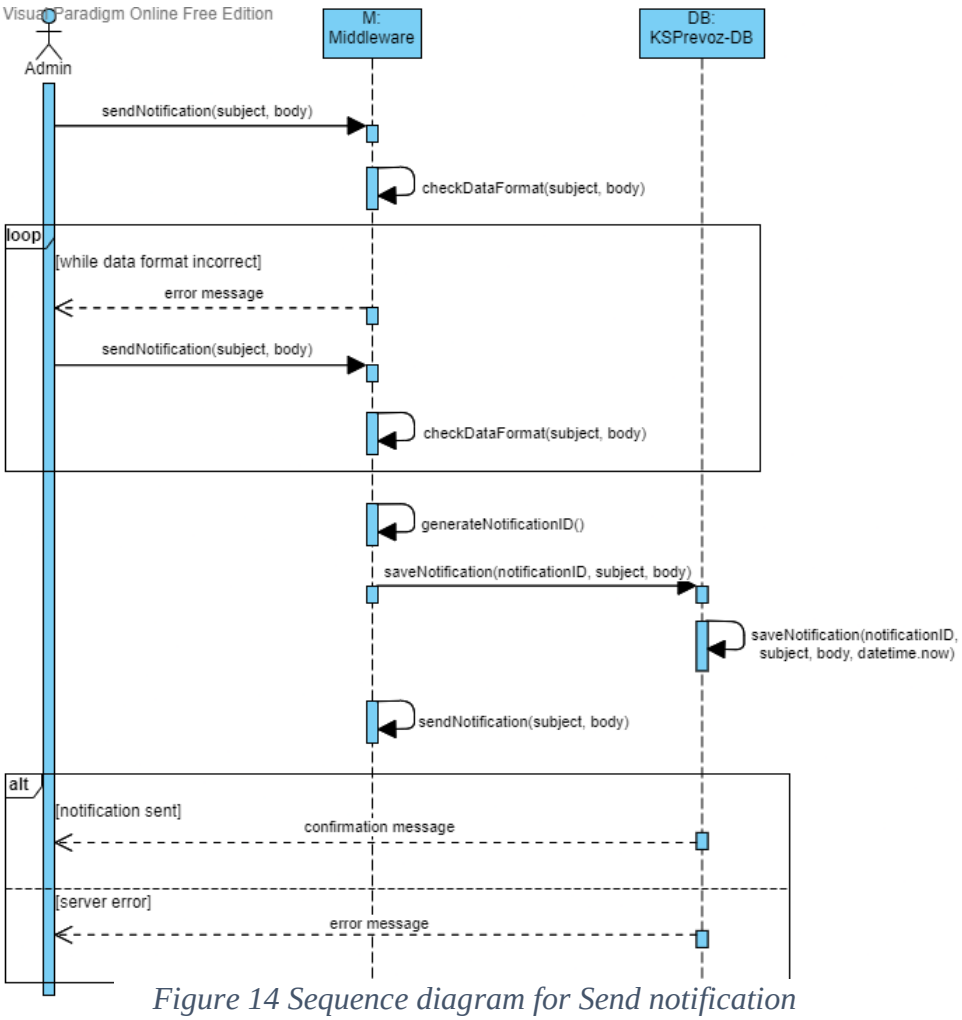

Figure 14 Sequence diagram for Send notification

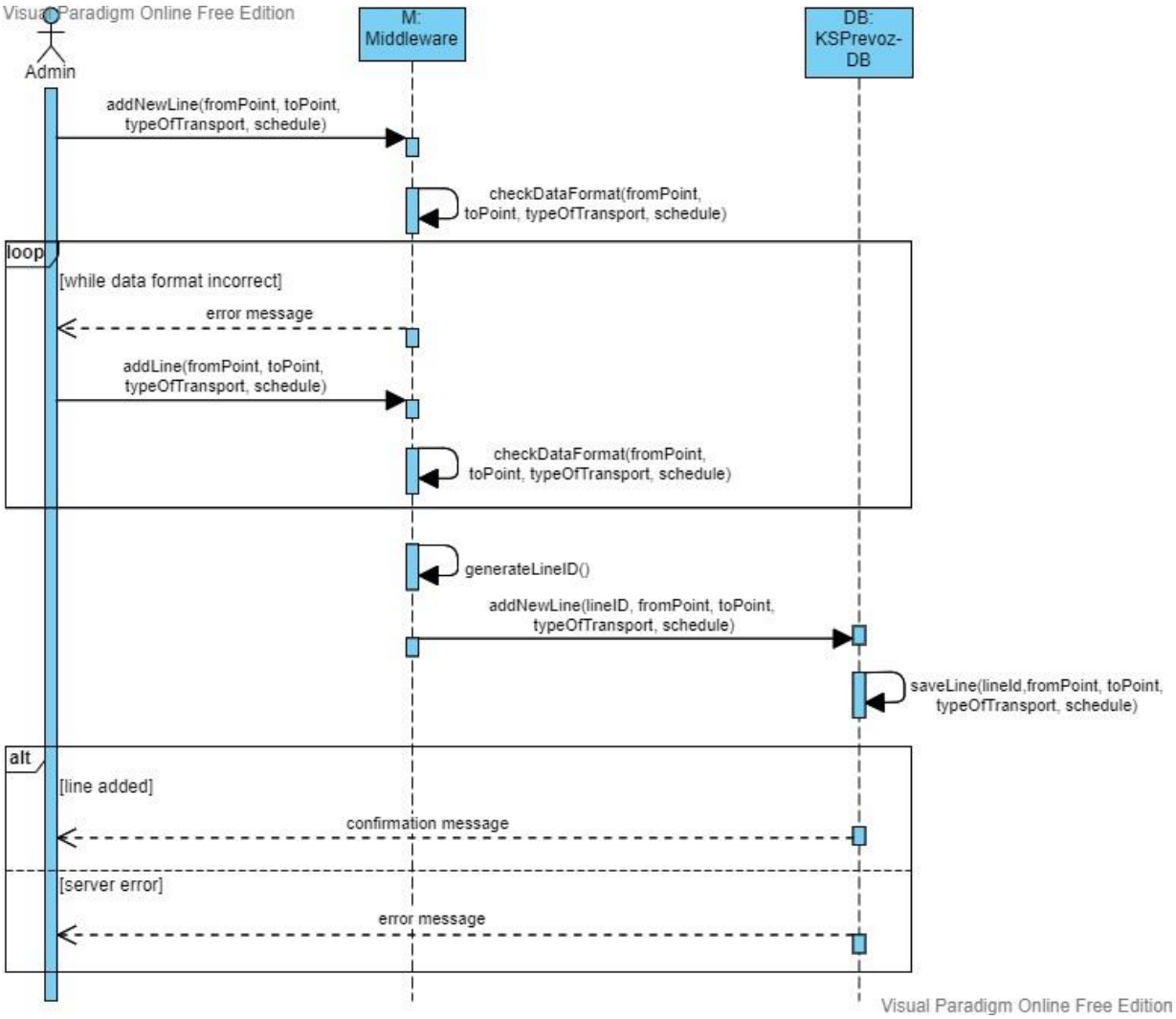

Figure 15 Sequence diagram for Add new line

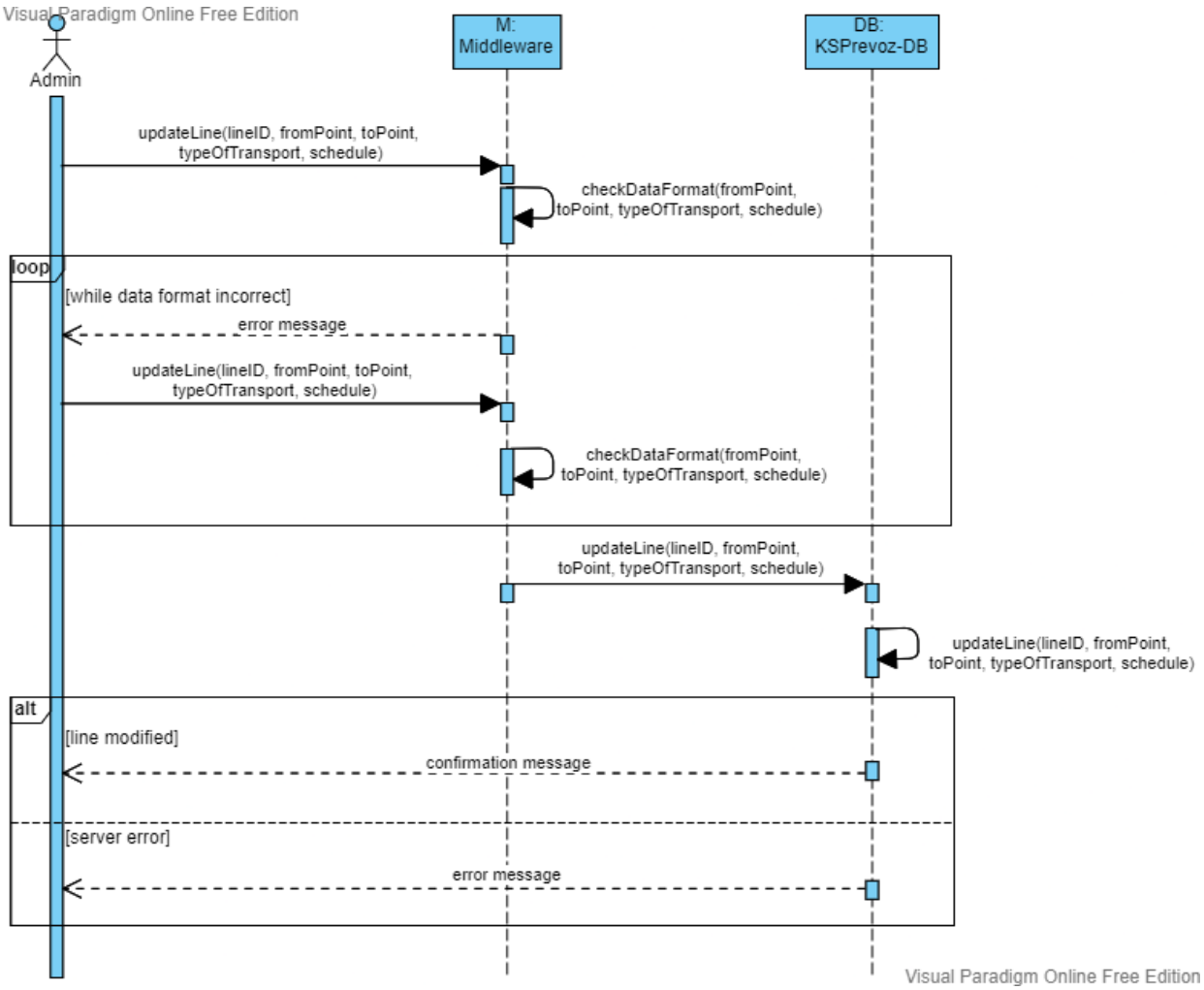

Figure 16 Sequence diagram for Update line

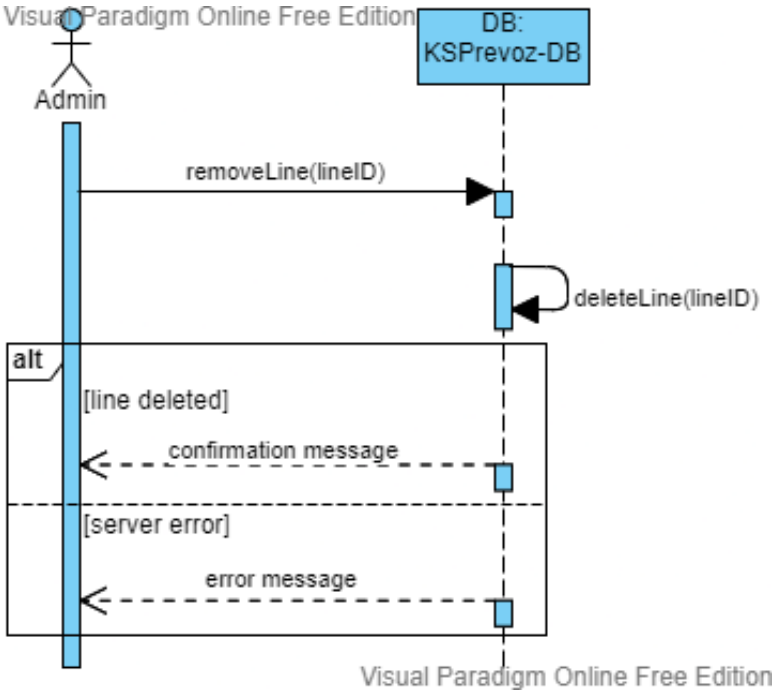

Figure 17 Sequence diagram for Remove line

2.5 User interface

For our user interface, we used *Figma* software to construct our mockup. We have chosen a variation of purple and yellow as our choice of colors because we found them not too tiring to look at for an extended period of time.

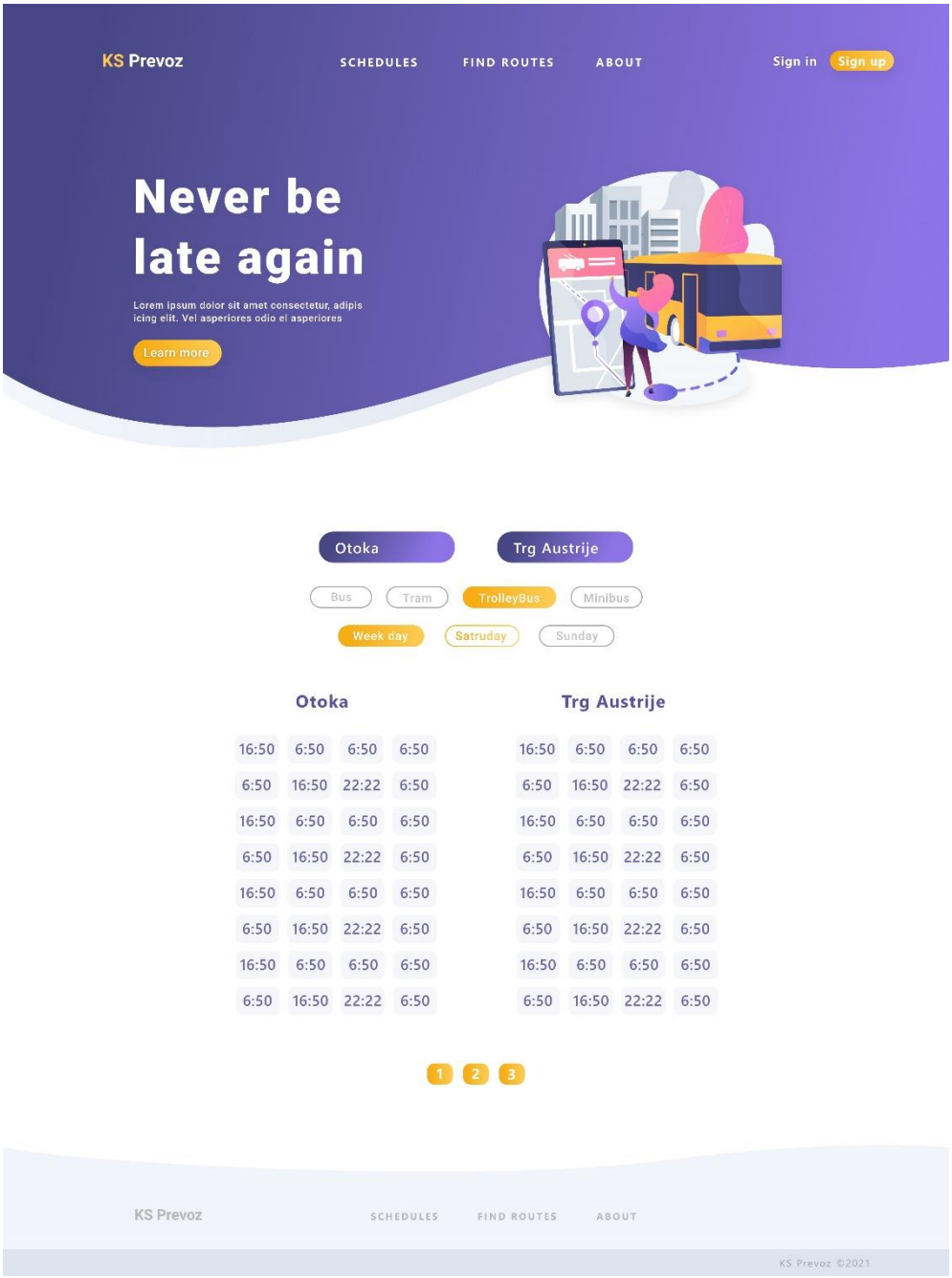

Figure 18 Landing page

Figure 18 refers to Landing page, main page of application. This page includes some general information about application, links to other pages, as well as section with “find schedule” feature. This feature will allow user to find schedule for specific line, type of transport and day of week.

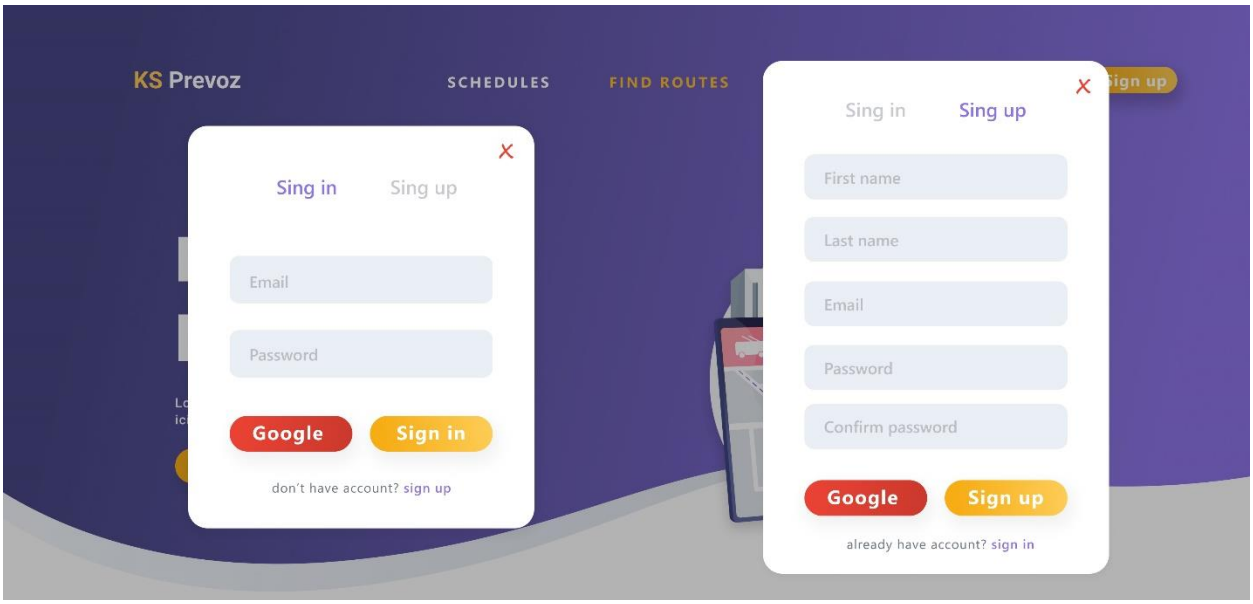

Figure 19 Sign in and sign up pop-ups

From navigation bar, if the user is not logged in, he will have the option to do so, and he will also have the option to Sign Up if he does not have an account. After clicking on the Sign Up or Log In button he them will be presented with a pop-up form for the corresponding operation. For the pop-up forms please refer to figure 19.

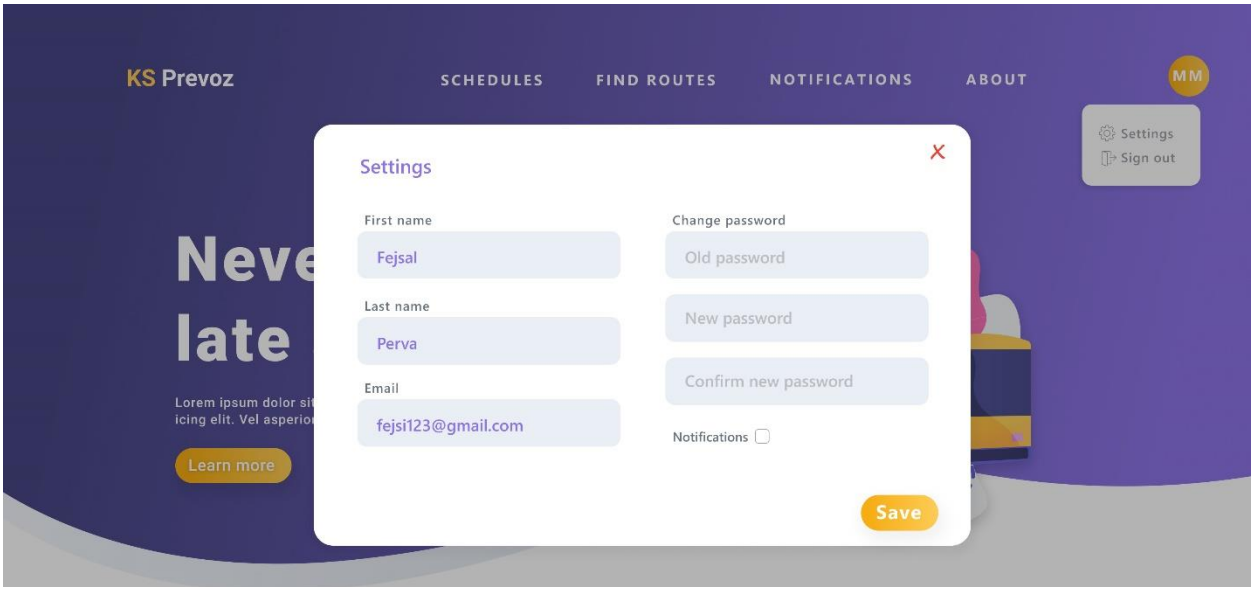

Figure 20 User settings

Logged in users will be able to change their account information if they wish to do so. Upon clicking the Settings button they will be presented with a pop-up form to input data that they wish to change. For account modification pop-up form refer to figure 20.

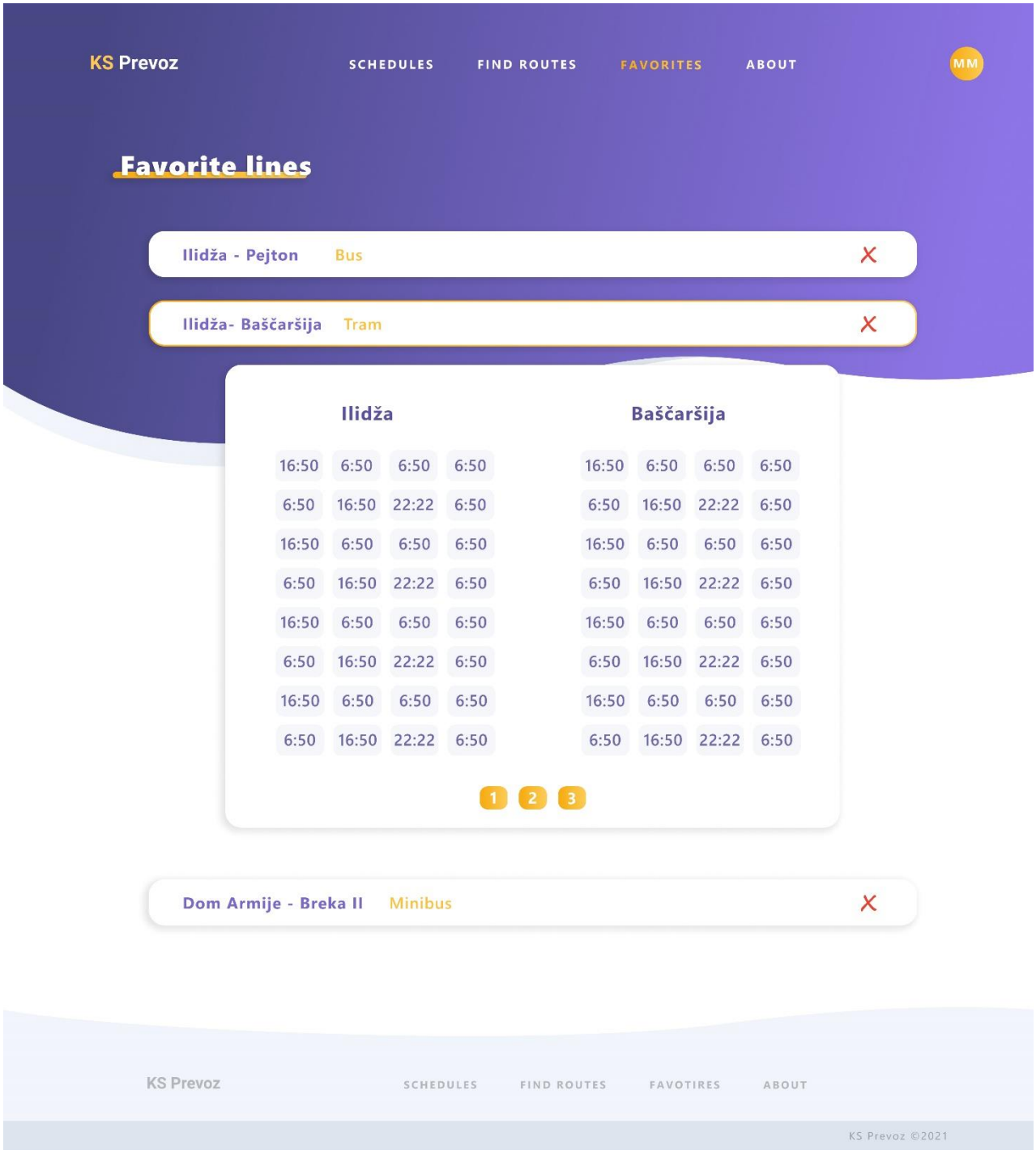

Figure 21 Favorites page

Favorites page will display a list of lines which registered users have marked as favorite. The main purpose is ease of access to lines which the user looks up regularly. The user has also the option to remove said lines from favorites list. We believe that these features will enhance user experience. For detailed UI design refer to figure 21.

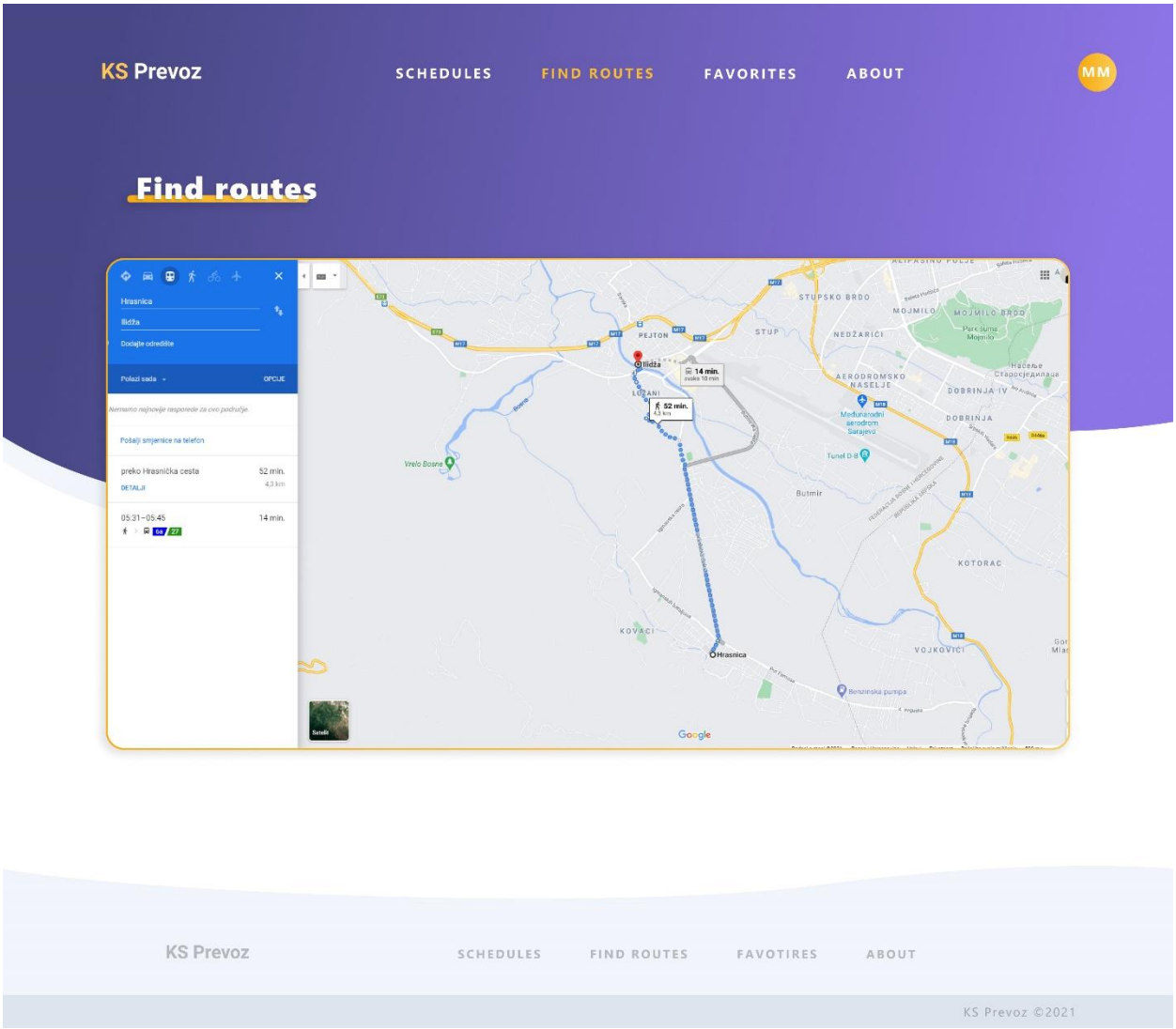

Figure 22 Find routes page

Find routes pages will provide users with the option to find routes between two points. The page will display a map and input fields for the user to input starting point and destination. Upon clicking "Search" the desired route will be displayed on the map as well as a list of available transport options for said route. For detailed UI design refer to figure 22.

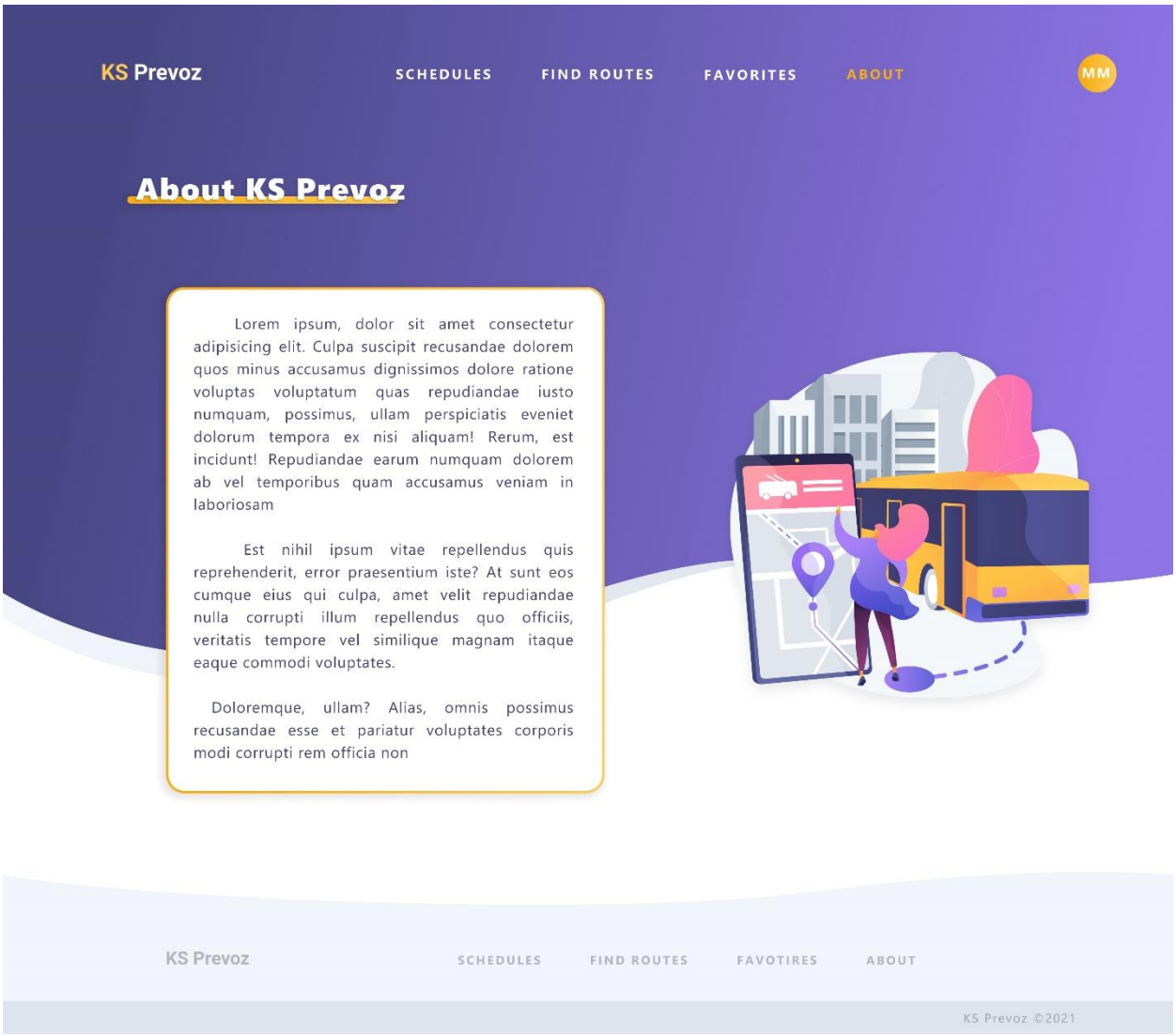

Figure 23 About page

About page will contain just some general information about KSPrevoz. UI of *about* page is visible in figure 23.

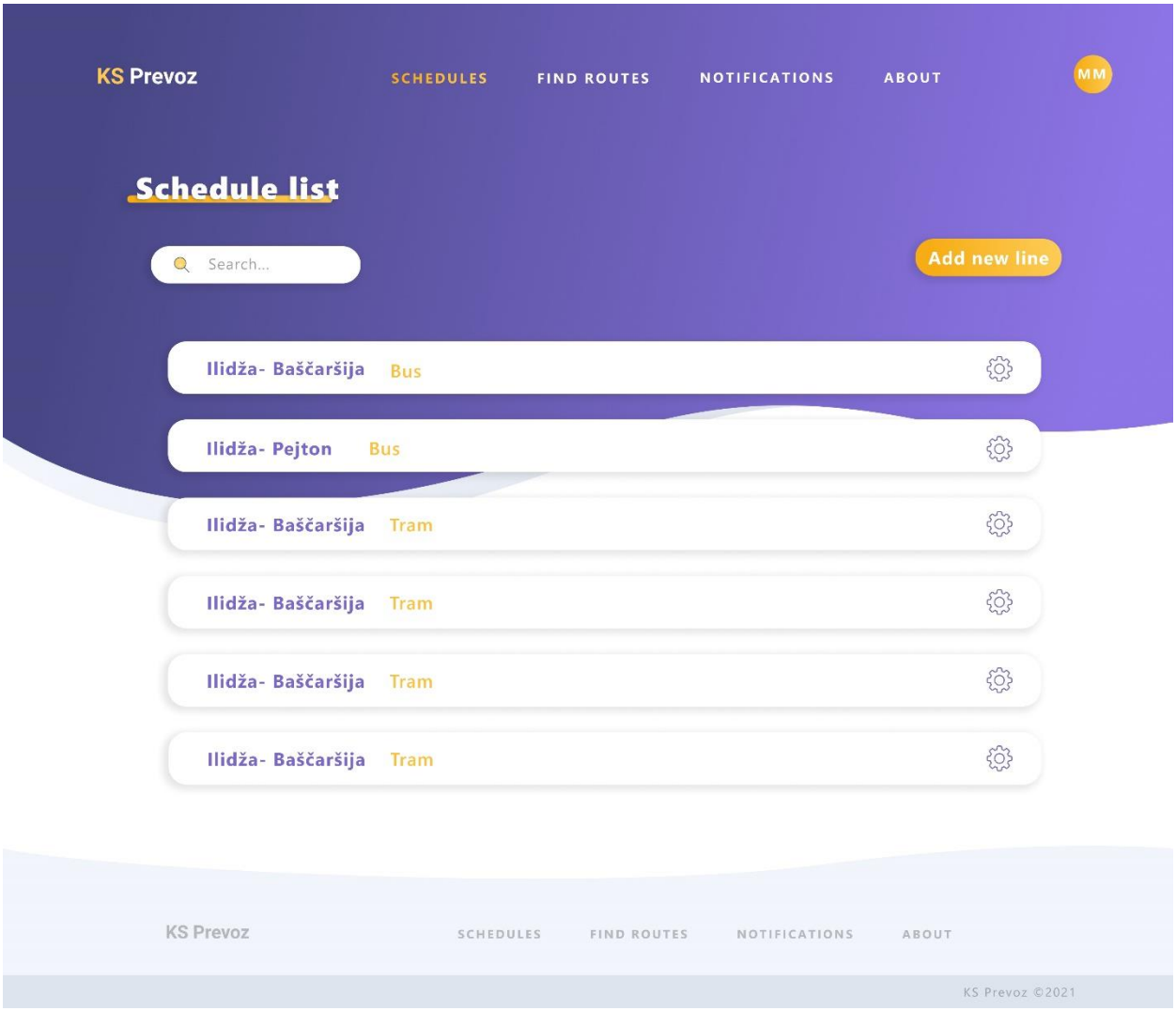

Figure 24 Schedule list

Schedule page for users with admin privilege will display a list of all lines available on the KSPrevoz website. The admin will be able to add a new line if he wishes. He will be able to do so by clicking on the "Add new line" button. Upon clicking, a pop-up form will be displayed where the admin will be prompted to input required data into the data fields. The admin will also have the option to search for a specific line via a search bar as well as to modify or delete a line. For the schedule page refer to figure 24 and for the popups for adding, deleting, and modifying a line refer to figure 25.

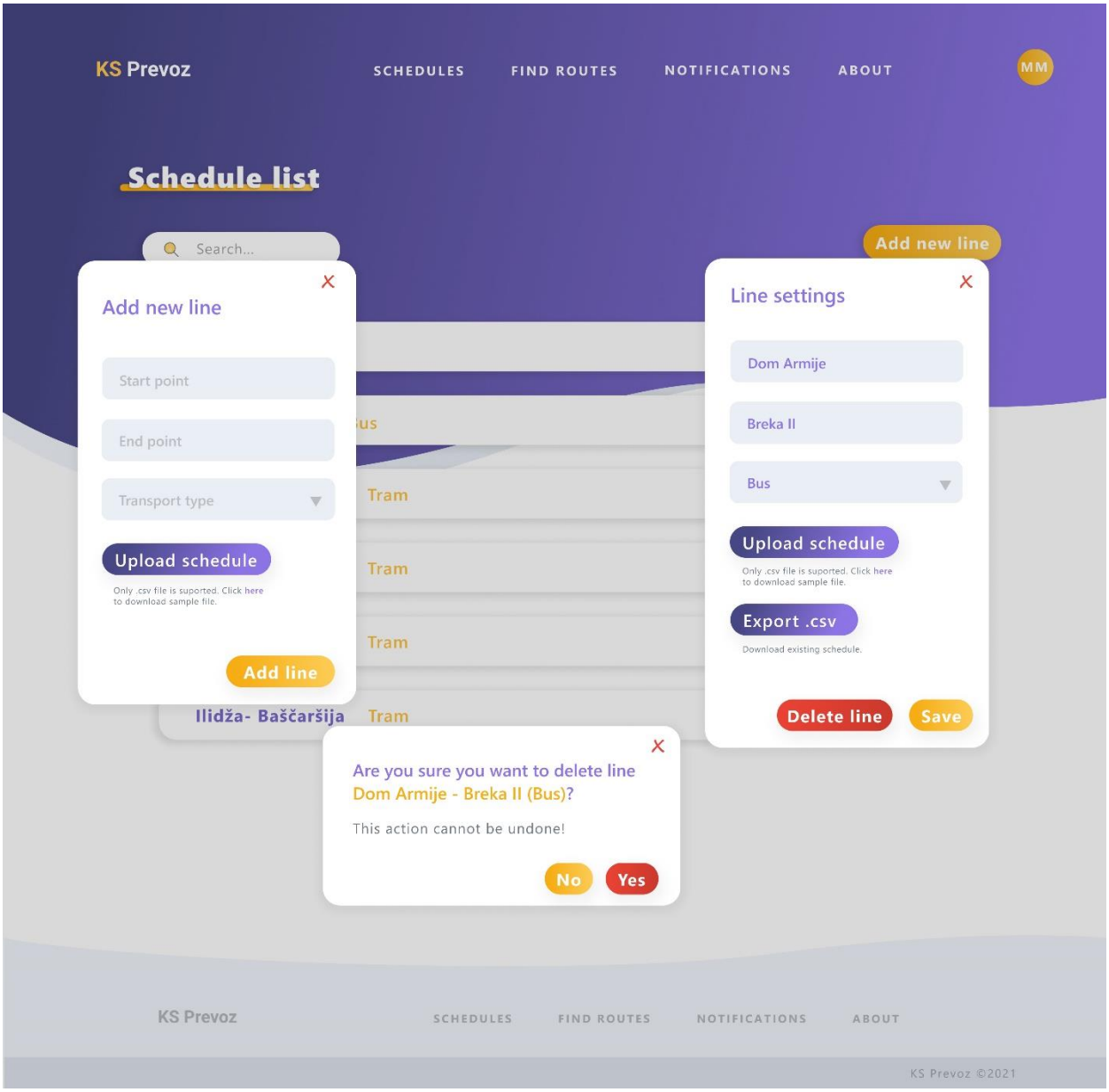

Figure 25 Popups for schedule list page

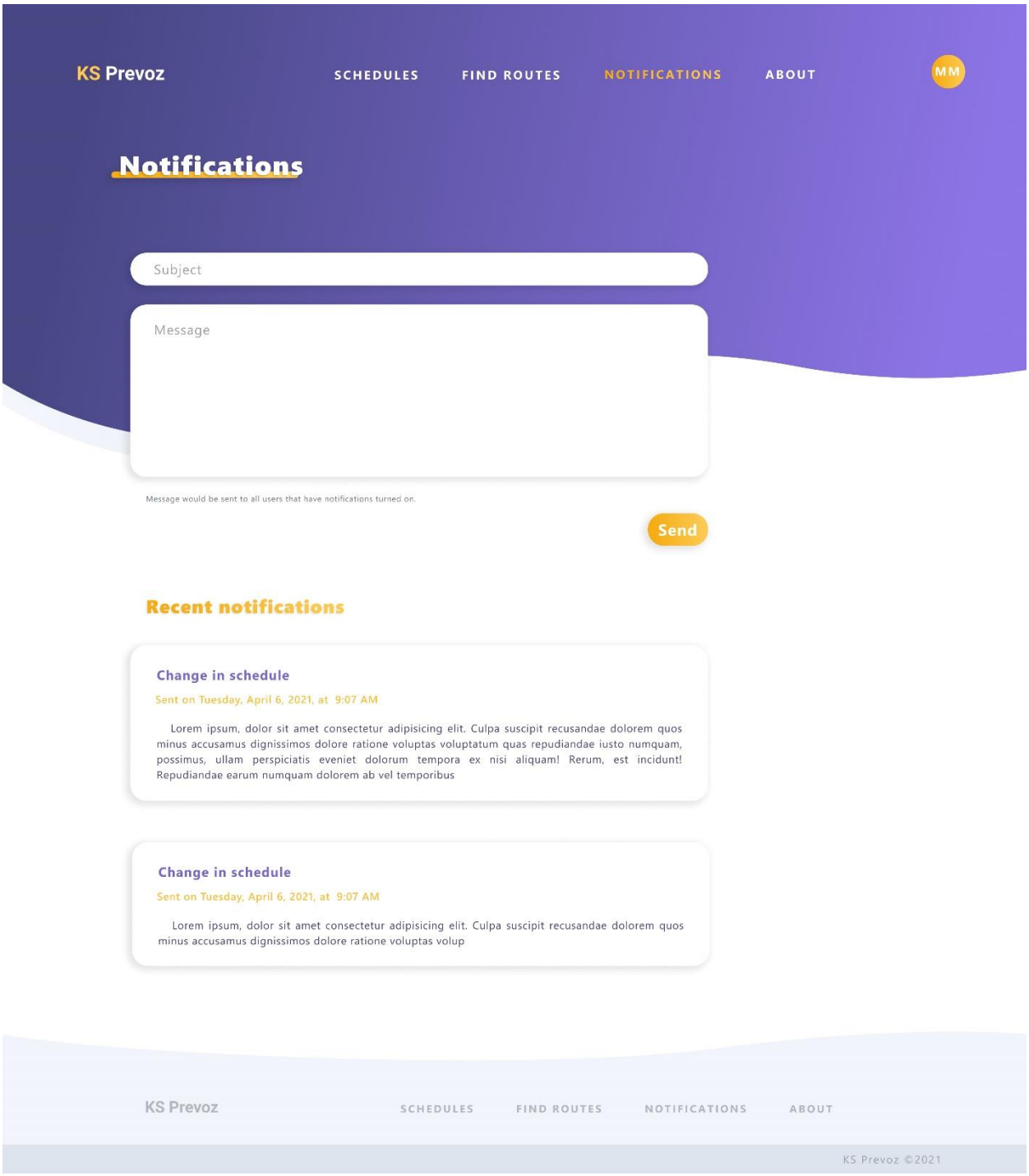

Figure 26 Notifications page

Notification page will enable users with admin privilege to send e-mail notifications to registered users about line schedule updates and any other information relevant to them. The page will also show a list of all previously send notifications.

Notifications page is visible in figure 26.

Contribution table

Since we did not have to make a lot of changes the whole team got in a Discord call, as usual, and together changed the Design document according to the feedback that we got from the professor for the first version of the Design document.

3 Implementation document

3.1 Contribution table

Team member	Tasks done
Harun Tucaković	Worked on backend. Implemented all authentication routes, user profile settings and 50% of routes in routes/lines.js. Deployed the backend of web application to the internet.
Muhammed Mušanović	Worked on frontend. Implemented authentication functionalities, favorite lines, schedule list, landing page, user profile settings. Did the design of the web application.
Fejsal Perva	Worked on frontend. Implemented find routes, find schedule, notification features, about page, and alerts, navbar and footer components.
Šejla Burnić	Worked on backend. Implemented all routes regarding notifications, favorite lines and other half of routes/lines.js. Web scraped gras.ba for schedules of public transport lines.

3.2 Trello link

https://trello.com/invite/b/k8YVCUgO/4907d43659340d34d272bfd16dccf9ae/ksprevoz

3.3 GitHub link

The code for this entire project can be found at the following GitHub repository (backend and frontend branches):

https://github.com/tucah1/KS-Prevoz

3.4 Deployment

Besides the finished code for our project we have also deployed the web application online for everyone to look at and use. The web application is deployed on Heroku.

You can access the KSPrevoz web application at: https://ksprevoz.herokuapp.com/