SRS DOCUMENT

Group 7, Project 13

February 2023

Contents

1 Introduction

The project's main objective is to build a portal for the **Cab Sharing** facility for the IIT Hyderabad residents.

1.1 Purpose

The document's purpose is to provide the requirements in the design, performance of the portal expected by the client. The document provides the brief details about the frameworks used, requirements of the user systems to access this portal and the functionalities it can perform.

1.2 Scope

The portal is designed to enhance the cab sharing facilities of the IIT Hyderabad residents.

1.2.1 In Scope

It allows the uses to provide the start point, end point, date, time of their journey to search or create a schedule group (merge) in the portal. The user can also view the details (Username, Phone number, Email ID) of already existing other users in a specific schedule group. All the users in a schedule group will be notified if a user joins/exits from their merge.

1.2.2 Out of Scope

The users cannot interact with each other inside the website. User cannot use their personal email IDs to register in the portal.

1.3 Definitions, Acronyms, and Abbreviations

The words **schedule group** and **merge** are used interchangeably, and they mean the same.
The word **User** includes all the residents of IITH with a active email ID.
Admin is given access to the entire DB and can perform CRUD operations on all the schedules.

1.4 References

1. Initial Requirements
2. Sample SRS provided in classroom
3. Online template

1.5 Overview

The requirements can change as the project proceeds with time. The terminology and definitions of the terms such as **merge, join, unjoin** might change in the future. The document will be updated accordingly.

2 Overall Description

2.1 Product Perspective

SRS is a comprehensive online platform where the users in the institute can access the platform with the required credentials. The platform's primary purpose is to provide the users with all the required information and make the ride efficient in terms of money and time.

2.1.1 System Interface

The HTML forms are used to take inputs from the user. We use NodeJS, Java to perform all the required operations in the system.

2.1.2 User Interface

Users can easily create, search, join, and exit schedules with a few clicks. First, all users will see the login page, where it asks username and password. After entering the correct credentials, user will be directed to the portal homepage (view page)where users can do various activities like create, search, join, and exit schedules.

2.1.3 Hardware Interface

a) **Server side**
For now, Our website will be hosted on a local host, port 3000.
b) **Client-side**
Our website will use standard hardware resources.(such as monitor, keyboard, mouse ,smartphones etc.)

2.1.4 Software Interface

a) **Server side**
MySQL database will be used for storing all the data about the system (for example: user information and schedules) in the database.We use NodeJS, Java to perform all the required operations in the system.
b) **Client-side**
The HTML forms are used to take inputs from the user. Any web browser which supports Java and HTML5 (like Google chrome) is needed.

2.1.5 Communication Interfaces

RestAPIs is used for communication(to store or retrieve data) between frontend and backend.

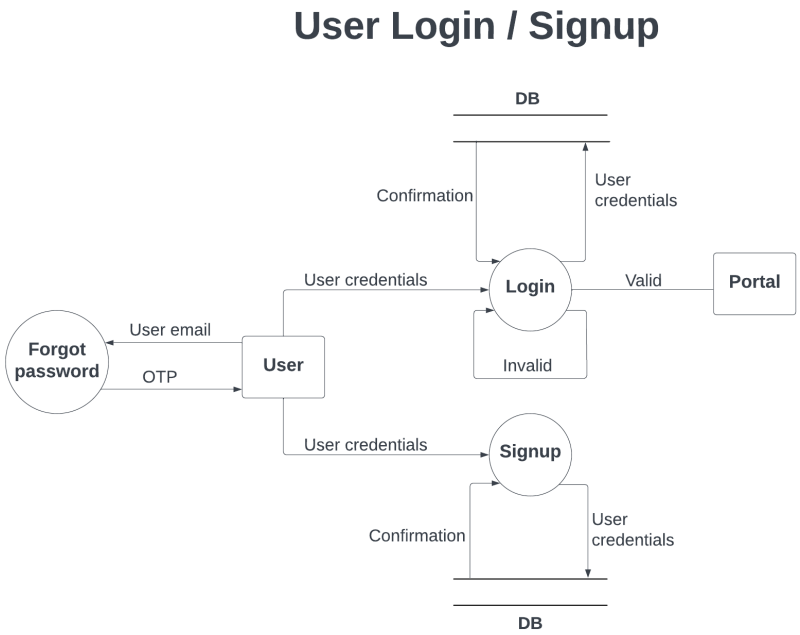
2.1.6 Memory constraints

1. Memory constraints occur in case of data overloading such as during high usuage from users during vacation times.
2. To avoid this, we should automatically remove the data for every one week or every two weeks after using the schedule.

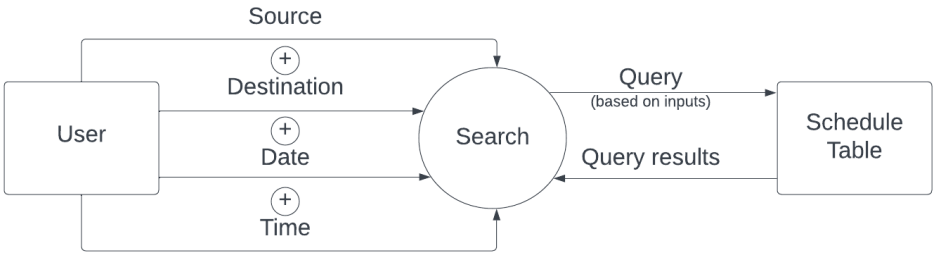
2.1.7 Operations

Certain operations should be implemented to protect the system from data failures such as database corruption, system failure, etc...

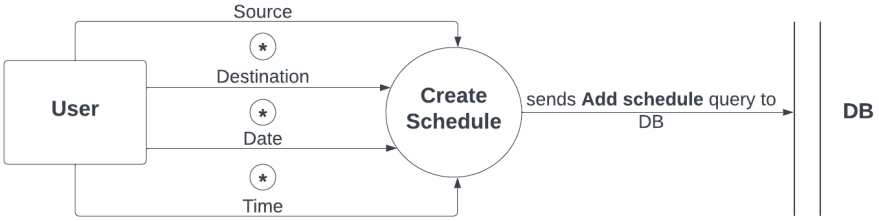
2.2 Product Functions



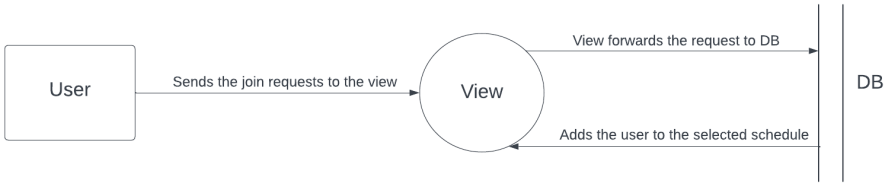
Search Function



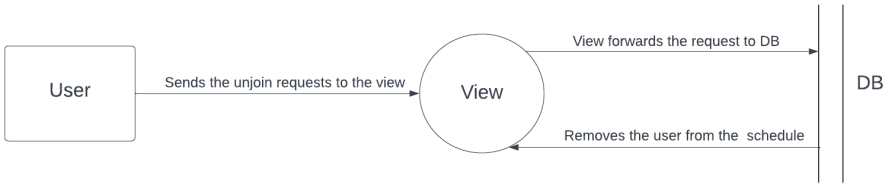
Creating Schedule



Join Schedule Function



UnJoin Schedule Function



2.3 User Characteristics

2.3.1 User

Residents of IIT Hyderabad are the primary users of the platform. They must first register/signup in to the portal to create an account. After successful login, they can create schedules or requests for cab sharing and read the schedules created by other users.

2.3.2 System Administrators

[!h] **Admin** is given all the access to the portal, can perform any action on the schedules created by the other users, can view the account details (except the password) of all the users, statistics such as the number of schedules created by each user, etc., can delete the account of the user or block him for using the portal in future in the extreme cases.

2.4 General Constraints

2.4.1 User Interface Constraints

2.5 Assumptions and Dependencies

1. We assume that the contact details provided by the user are valid and correct, which are further used by the group members for a better travel discussion.

3 Specific Requirements

3.1 Performance Requirements

1. The system is expected to handle around 300 users simultaneously.

2. Since it is a portal used by university students, sometimes, during vacation holidays, there is a chance of an overload of operations from the users. So it is expected to handle at least 300 users at a time

3.2 Logical Database Requirements

All the data (user profile details, upcoming and past schedule details) are stored in the database. The database is made to allow concurrent access to have friendly access.

3.3 Design Constraints

1. The user view (front end) is designed using React.
2. The front end integrates with the database by using SQL.
3. Frontend and backend communicate via API requests.

3.4 Software System Attributes

This software comprises of:

1. ReactJS application for user view
2. NodeJS for backend
3. MySQL as database

3.4.1 Reliability

1. In case of wrong credentials, an error message would be displayed without any cause of system failure.
2. Since it is currently used locally, the system would be reliable towards errors due to more concurrent users.

3.4.2 Availability

1. The system should be made available all the time.
2. In case of any data transmission failures, such as database corruption, the data in the form of backups will be retrieved and saved using the MySQL server.

3.4.3 Security

1. The users' login credentials are stored in the database in an encrypted format.
2. The past schedules of the user are kept private to the user only and are not shown as public.

3.4.4 Maintainability

The database is maintained by MySQL, and re-initialization of the program is implemented in case of any failure in the database.

3.4.5 Portability

1. The web application will be windows-based and will be made compatible with other architectures too.

2. The frontend, backend, and database applications will be independent of the system and will be made compatible with any web browser.

4 Use Cases for the functionalities

4.1 SignUp

Primary Actor: User
Pre-Condition: No pre-conditions required (start point).
Main Scenario:
1. User provides the email ID, name, phone number, and password.
2. The user has to re-enter the password for verification.
3. The user details are then stored in the DB, and the account is created for the user.

4.2 Use cases related to system authorization

4.2.1 login

Primary Actor: User
Pre-Condition: The user has an account (signed up already with the portal).
Main Scenario:
1. User provides the user ID and password.
2. The portal does the authentication and allows him/her to log in succesfully to proceed further.
Alternative Scenario:
1. If the entered password is incorrect, the user will be redirected to the login/signup/forgot password page.

4.2.2 forgot password

Primary Actor: User
Pre-Condition: The user has an account (signed up already with the portal).
Main Scenario:
1. The user is asked to provide the email address or phone number they used to register for the portal.
2. The user then receives a One-Time-Password, which he/she has to enter into

the portal.
3. Each OTP is valid for 15 minutes. 4. For a successful entry of the OTP, the user can now set a new password. Alternative Scenario:
1. If the user didn't register for the portal yet, he/she will be notified to sign up for the page first. 2. For an unsuccessful entry of the OTP or entering a wrong OTP, the user will receive a notification about the attempt. 3. The user can request another OTP after 15 minutes again.

4.3 Use cases for profile

4.3.1 edit profile

Primary Actor: User Pre Condition: User logged in successful Main Scenario:
1. User initiates on "edit" functionality.
2. User can use this functionality for name, phone number .
3. User enters new details and clicks on save.
4. profile is changed in the database and the display.

4.3.2 delete profile

Primary Actor: User
PreCondition: User logged in
Main Scenario:
1. User Initiates the delete functionality
2. System asks for the password for profile
3. Password entered is verified if entered correctly, the profile is deleted.
Alternative Scenario:
1. If the entered password is incorrect profile will not be deleted.

4.4 Use cases for schedule

4.4.1 view schedule

Primary Actor: User
Pre-Condition: User logged in
Main Scenario:
1. The user can view all the pre-existing schedule groups (merges) created by the other users and can view the members present in them.
2. The user can create a new schedule group of his own or can join any of the other pre-existing merges.
3. The user is also provided a search bar to filter out the existing schedules.

4.4.2 search schedule

Primary Actor: User
Pre-Condition: User logged in

Main Scenario:
1. The user can search among the pre-existing schedules by giving a start point and/or end point (destination) and/or travel date and/or travel time.
Alternative Scenario:
1. If the search bar is empty, the pre-existing schedules are sorted out so that the schedules of which the user is already a part are shown at the top, and others are shown in ascending order of time when they are created.

4.4.3 create schedule

Primary Actor: User
Pre Condition: User Logged in
Main Scenario:
1. User enters the source, destination, date, and time in the search bar.
2. User clicks on the Create button.
3. New schedule will be created and added to the list of existing Schedules.
4. User will be joined into the schedule group which he/she created with **count** as one and **status** as joined with source, destination date time.
Alternate Scenario:
1. If the schedule is already created by him with the same date, same time and same destination, and same source then the creation will be failed and shows that already exist.

4.4.4 join schedule

Primary Actor: User
Pre Condition: User Logged in
Main Scenario:
1. User can see the pre-existing schedules created by other users from view.
2. User can use the search bar for filtering based on date, time, source, and destination.
3. User can click on **join** button to enter into a schedule group (merge).
4. Count for that schedule increments when a user joins it.

4.4.5 unjoin or exit a schedule

Primary Actor: User
Pre Condition: User Logged in
Main Scenario:
1. User can see the schedules in which he/she is a part of, in **view** page.
2. User can use the search bar for filtering based on date/time/source/destination.
3. User can select on **unjoin** button (as soon as a user joins a schedule group (merge), the **join** button becomes **unjoin**).
4. User will be unjoined from the schedule group.
5. Count for that schedule decreases by 1.