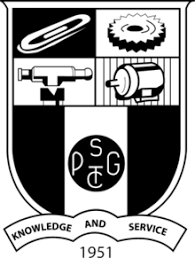
**PSG COLLEGE OF TECHNOLOGY**

**Department of Information Technology**

****

**15I404 Database Management Systems**

**Project Title: Airline Reservation System**

**Name: Shwethaa R**

**Roll Number : 18I351**

1. **Introduction**

In our present day and age, almost every single one of our everyday actions are carried out via the internet and technology. The project that we have chosen is an airline reservation system. It is an incredibly tiring and exhausting process to book flight tickets physically. It takes time and effort. It is also quite hard to manage all the details keeping track of all the bookings.

Airline reservation System is a computerized system used to store and retrieve information and conduct transactions related to air travel. The project is aimed at exposing the relevance and importance of Airline Reservation Systems. It is projected towards enhancing the relationship between customers and airline agencies through the use of ARSs, and thereby making it convenient for the customers to book the flights as when they require such that they can utilize this software to make reservations.

Our project on airline reservation system helps to simplify this process greatly. We can easily book our flights online, keep track of booking and even reschedule and cancel flights. Thereby simplifying the whole process and saving valuable time. The NoSQL database that we have chosen is MongoDb. Our front end is done with the help of HTML, CSS and Javascript. Our backend development is done with the help of Node js.

1. **Existing System**

In all the existing airline reservation systems, there is an existing flight booking system available in branch. The system is run manually by the database. It is difficult to run efficiently by manpower, and difficult to respond every user within a short period. Thus, we come to know the essential need to make its web application.

In some cases, it is possible to book flight tickets online but the only way possible is via a third party. This system does not permit the user flexibility and freedom to book their flights.

Since the third party is a middle man, it may also cause the user to incur additional charges

**Drawbacks of Existing System:**

* It is difficult to run efficiently by man power.
* The system is run manually by the database.
* It is difficult to respond every user within a short period.
* The danger of losing the information in some cases.
* The problem of maintaining the information of registered users.
* Cost is more in case of third party applications

1. **Proposed System**

In our proposed system we plan to give the user freedom as well as flexibility. Our system is simple and pleasing enabling everyone to use it. The following are the proposed features in our usecase of an airline reservation system

|  |  |
| --- | --- |
| 1. | Enable user to login with the help of their registered phone number |
| 2. | If the user does not have a registered account, they are provided with an option to signup for a new account |
| 3. | User has an option to view all their user details |
| 4. | User has an option to view all their flight bookings so far |
| 5. | User has an option to book a new flight |
| 6. | Users have an option to view all flights when they provide the departure as well as the destination |
| 7. | Users have an option to cancel their flights |
| 8. | Users have an option to also reschedule their flights |

1. **Need for the system**

An airline reservation system can be of great use and importance to the people. A few of the reasons why an airline reservation system is needed is elaborated below

|  |  |
| --- | --- |
| 1. | An airline reservation system enables users to book flights with ease |
| 2. | It reduces the manual labour required to book flights each time |
| 3. | This system reduces the chances of double booking |
| 4. | This system almost completely eliminates the chances of a manual error |
| 5. | This system makes the administration work easier |
| 6. | It helps users easily reschedule their flights |
| 7. | It helps users easily cancel their flights |
| 8. | It helps to keep track of all the flights, their details and the number of seats in each flight |

1. **Contribution to the project**

With regards to our project of an Airline Reservation System, I, Shwethaa R(18I351) have made the following contributions.

* I wrote the code for the rescheduling of flights and to view and update user details
* I helped to design all of our front end pages
* I helped out many of my team mates by helping them understand the backend of node js more.
* I contributed to removing bugs and helping my team mates correct their errors in their respective code

1. **About our frontend and backend**

The front end we have chosen for our project is HTML, CSS and JavaScript.

The NoSQL database backend we have chosen is MongoDB.

We have used Node.js for our backend development

* 1. **About the frontend**
* **HTML:**

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. It is constantly undergoing revision and evolution to meet the demands and requirements of the growing Internet audience under the direction of the W3C, the organisation charged with designing and maintaining the language.

HTML consists of a series of short codes typed into a text-file by the site author — these are the tags. The text is then saved as a html file, and viewed through a browser, like Internet Explorer or Netscape Navigator. This browser reads the file and translates the text into a visible form, hopefully rendering the page as the author had intended. Writing your own HTML entails using tags correctly to create your vision. You can use anything from a rudimentary text-editor to a powerful graphical editor to create HTML pages. The tags are what separate normal text from HTML code. You might know them as the words between the . They allow all the cool stuff like images and tables and stuff, just by telling your browser what to render on the page. Different tags will perform different functions. The tags themselves don’t appear when you view your page through a browser, but their effects do. The simplest tags do nothing more than apply formatting to some text

* **CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics. It enables multiple pages to share formatting and reduce complexity and repetition in the structural content (such as by allowing for table less web design).

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the author or the reader did not link the document to a specific style sheet the default style of the browser will be applied.CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

* **Javascript**

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications. JavaScript is a prototype-based scripting language with dynamic typing and has first class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multiparadigm language, supporting object-oriented, imperative, and functional programming styles. The application of JavaScript to use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but just-in-time compilation is now performed by recent (post-2012) browsers.

Javascript can be used along with HTML and CSS for form validations.

A JavaScript framework can be a powerful tool you can use to help render the page. These are typically only used when there are complex dynamic interactions that need to occur.

* 1. **About the backend**
* **MongoDB**

MongoDB is an open source database management system (DBMS) that uses a document-oriented database model which supports various forms of data. It is one of numerous nonrelational database technologies which arose in the mid-2000s under the NoSQL banner for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents.

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JavaScript Object Notation objects but use a variant called Binary JSON (BSON) that accommodates more data types. The fields in documents are akin to the columns in a relational database, and the values they contain can be a variety of data types, including other documents, arrays and arrays of documents, according to the MongoDB user manual.

Documents, which also must incorporate a primary key as a unique identifier, are the basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables. Collections can contain any type of data, but the restriction is the data in a collection cannot be spread across different databases.

The mongo shell is an interactive JavaScript interface to MongoDB which allows users to query and update data, and conduct administrative operations. The shell is a standard component of the open source distributions of MongoDB. Once MongoDB is installed, users connect the mongo shell to their running MongoDB instances.

The BSON document storage and data interchange format used in MongoDB provides a binary representation of JSON-like documents. Automatic sharing is another key feature that enables data in a MongoDB collection to be distributed across multiple systems for horizontal scalability as data volumes and throughput requirements increase.

The NoSQL DBMS uses a single master architecture for data consistency, with secondary databases which maintain copies of the primary database. Operations are automatically replicated to those secondary databases for automatic failover.

* **Node.js**

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

**Features of Node.js**

Following are some of the important features that make Node.js the first choice of software architects.

* **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks

**In our project, we use Node.js in addition to some middle-ware modules :**

These modules are:

* Express (require(‘express’))
* Express-session (require(‘express-session’))
* Path (require(‘path))
* Express-handlebars (require(‘express-handlebars’))
* Mongoose (require(‘mongoose))
* Body-parser (require(‘body-parser))

1. **Reason for choosing this backend**

The backend NoSQL database used in MongoDB and Node.js

The reason why we chose MongoDB over other NoSQL databases are :

* 1. MongoDB is one of the **fastest growing** and very popular databases and is ranked 5th on the world’s most popular databases
  2. MongoDB has a very **flexible data model**. Since MongoDB does not have any internal schema, we could easily change the data model in our project at any given time.
  3. MongoDB documents supports all data types.
  4. MongoDB has drivers for almost every language and therefore it was easier to implement it with Node.js.
  5. Since MongoDB is document oriented, it is easy to integrate it with other applications. MongoDB uses JSON (JavaScript Object Notation) based document storage schema called BSON.
  6. Since we are already studying about MongoDB in our DBMS class, it is comparatively easier to implement than the other NoSQL databases.

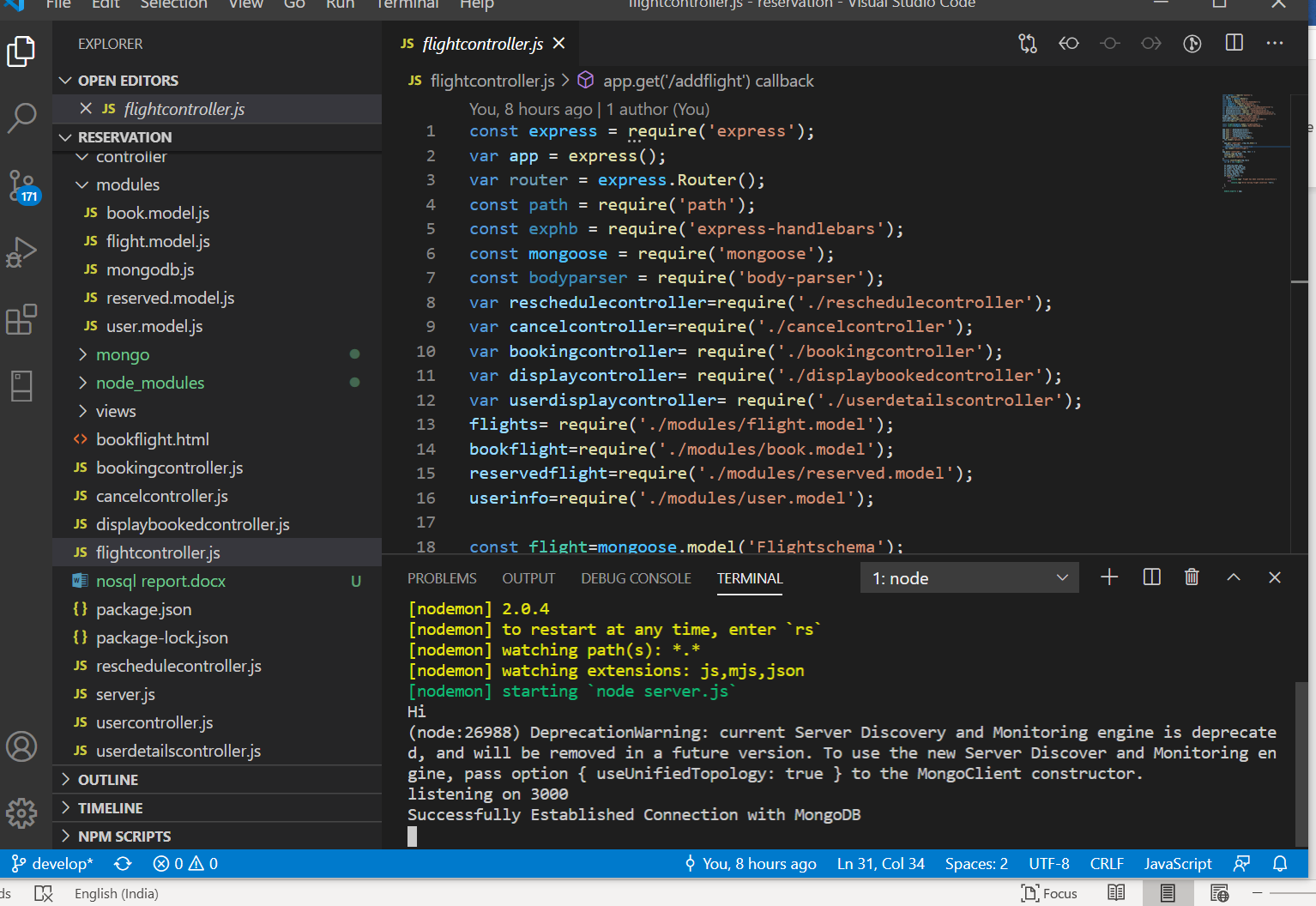
The reasons why we chose Node.js over other backend developers like php are:-

* 1. It is faster to develop apps with the help of Node.js
  2. Coding with JavaScript can be done with both the client side and the server side
  3. We are able to use the npm package (Node Package Manager) that helps us by giving us many middle ware modules to help with the development process.

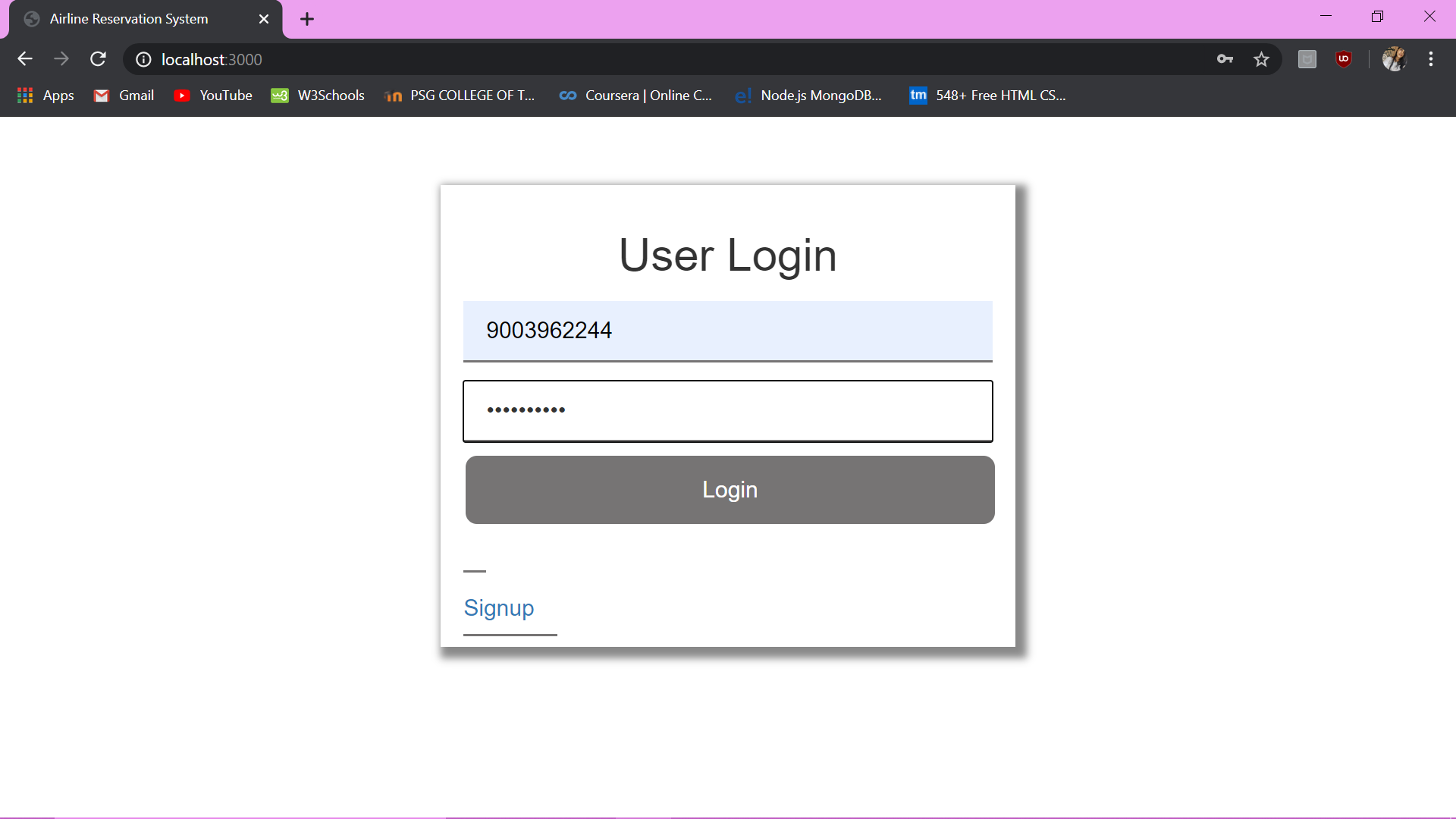
For example, in our project we use npm modules such as express, path, mongoose, express-handlebars, express-sessions

* 1. We can open, read and delete files on the server side
  2. It was easier to learn
  3. We are able to keep the JSON format in our database
  4. It helps us share code between the browser and the backend
  5. Node.js has very good scalability
  6. When we use Node.js along with VSCode, we can simultaneously debug code on the browser and Node code

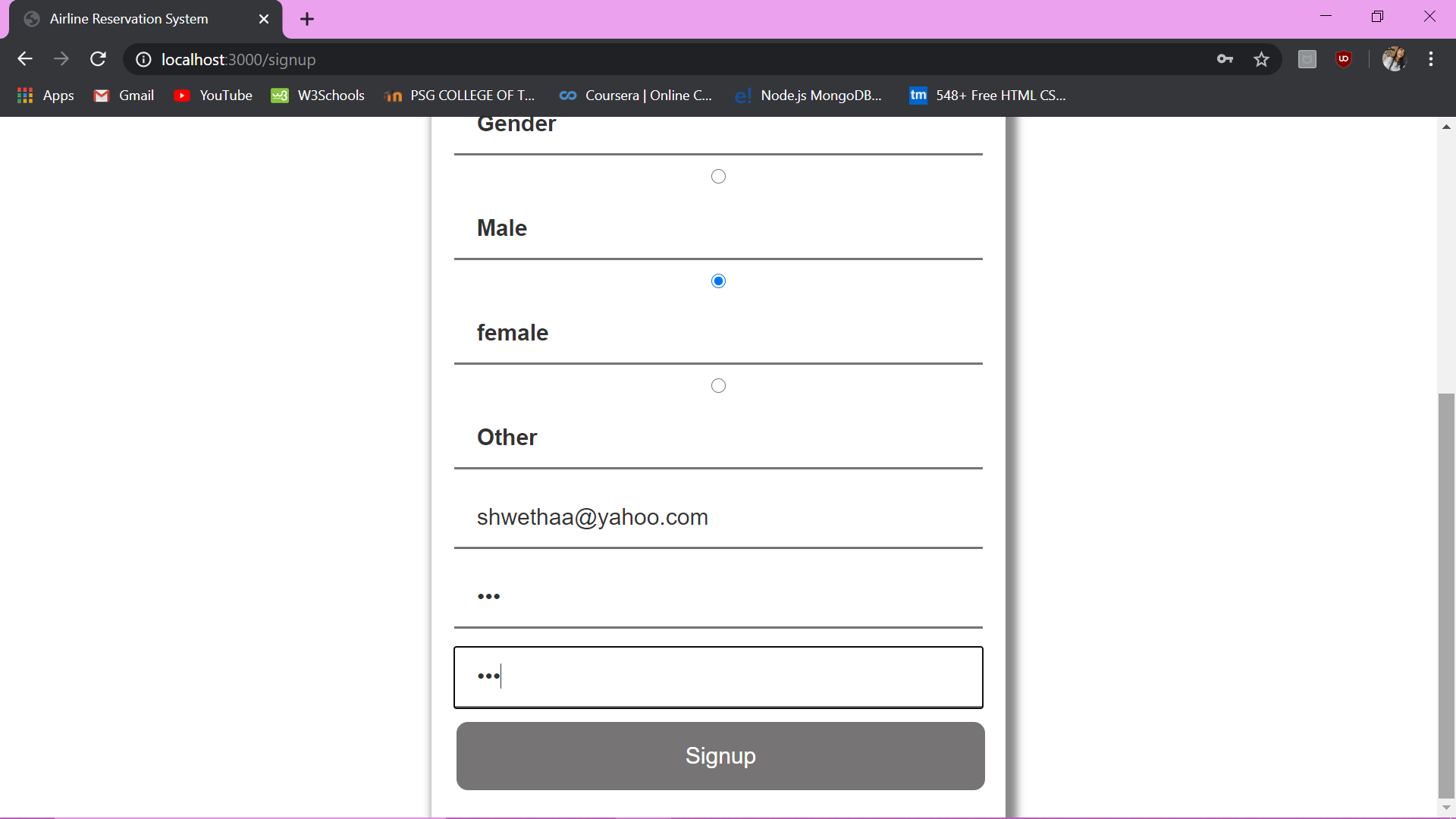
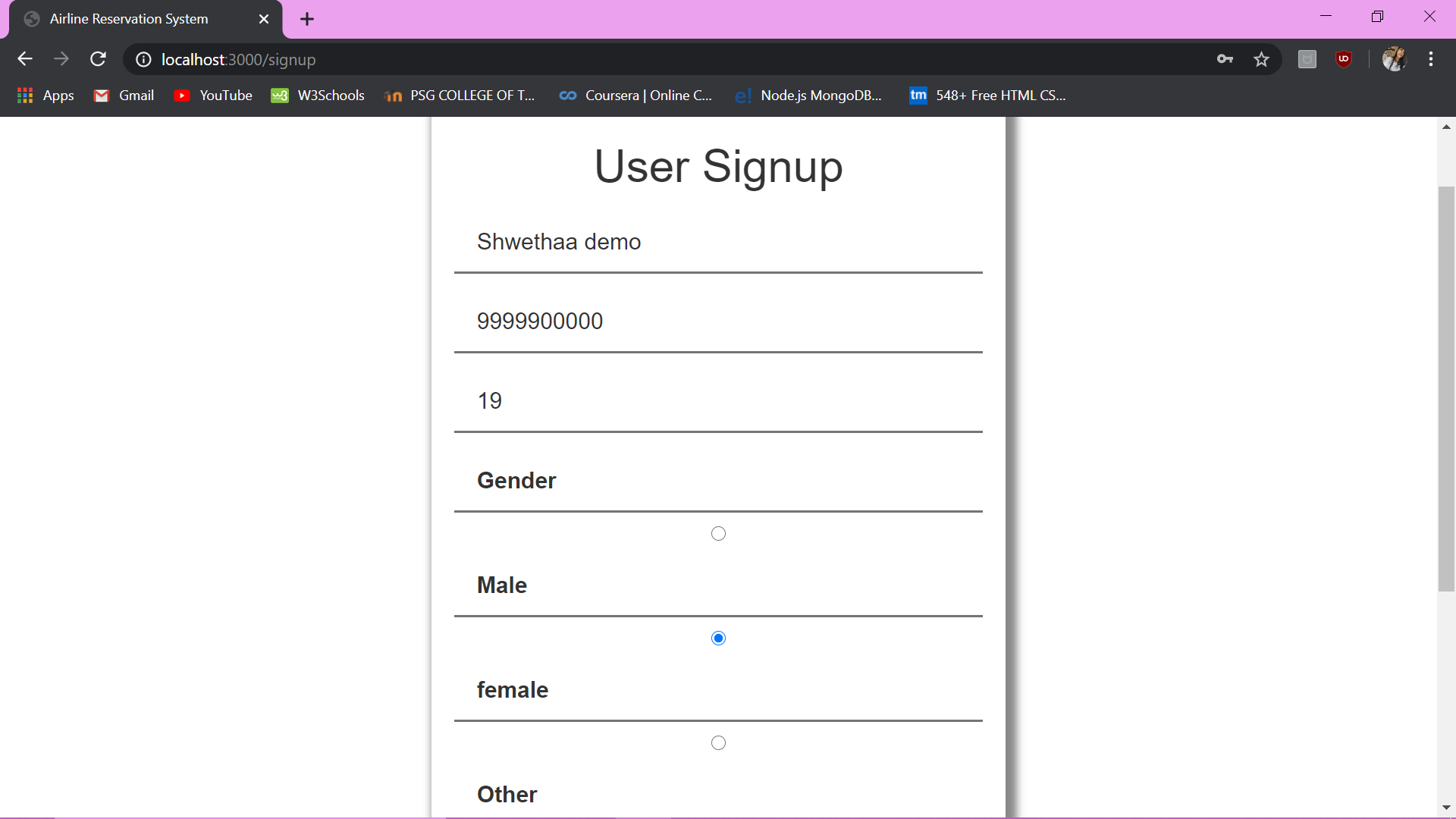
1. **Screenshots of the output:**

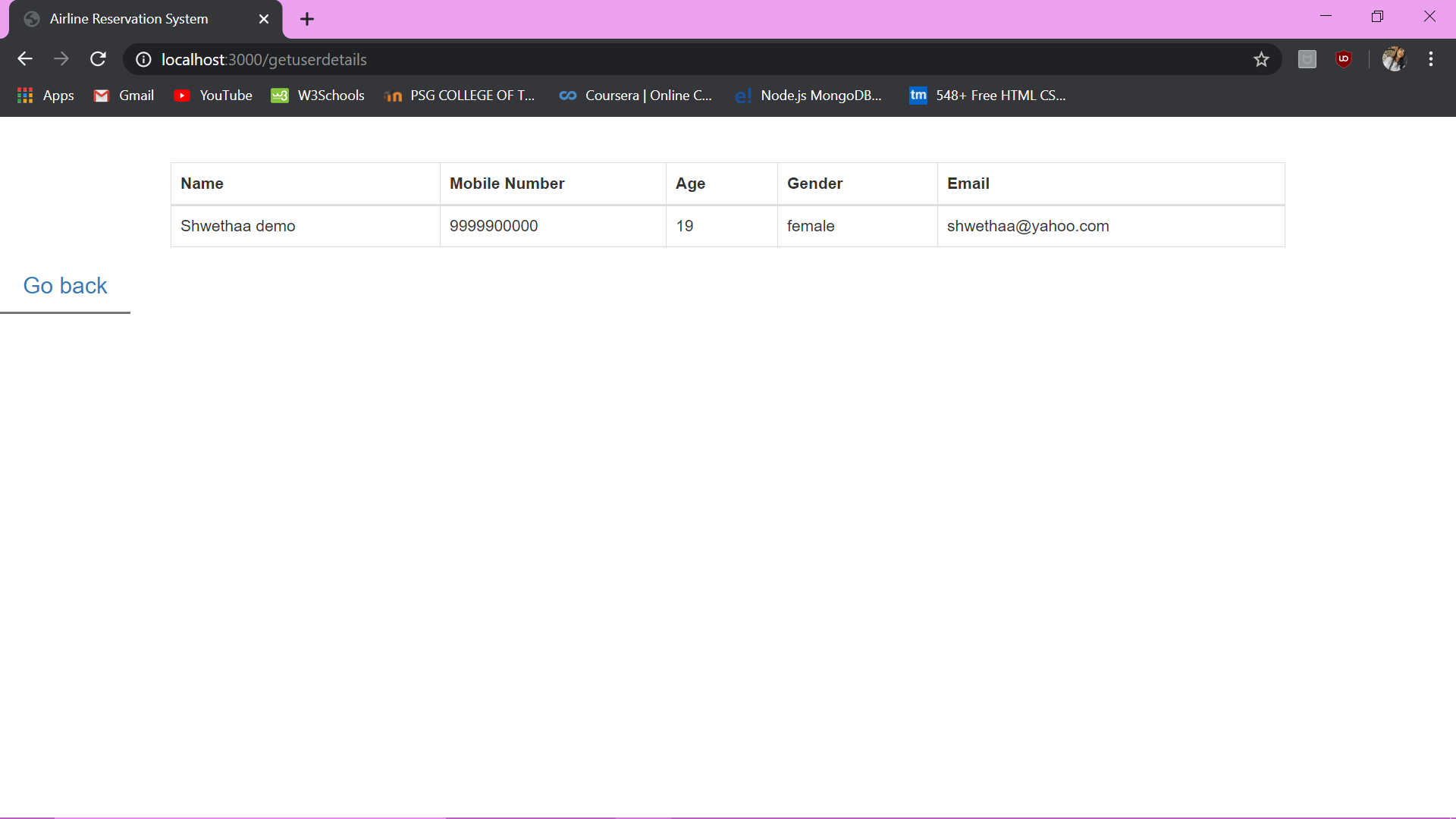
****

Our homepage/ User Login:

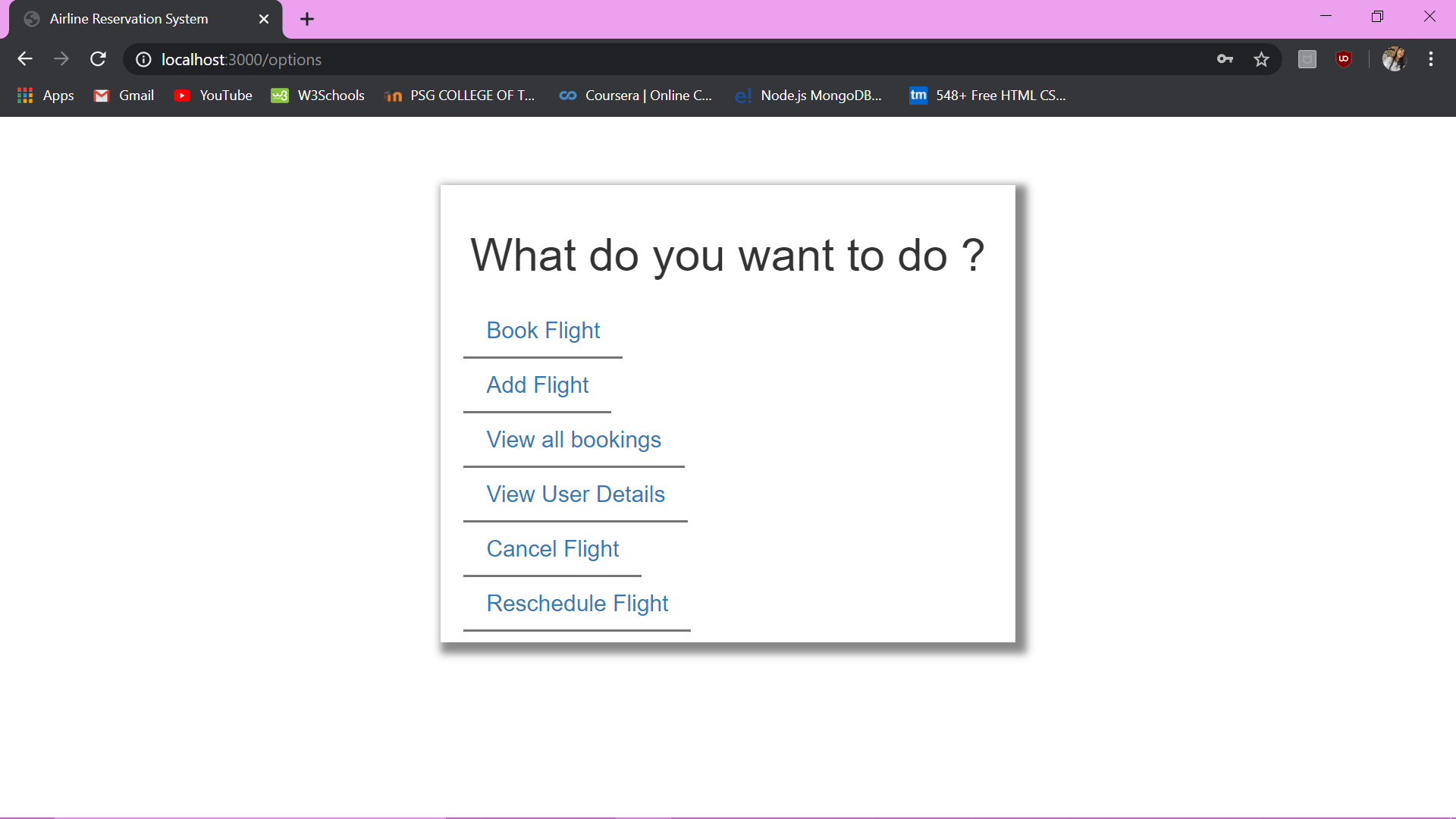
****

Signup page:

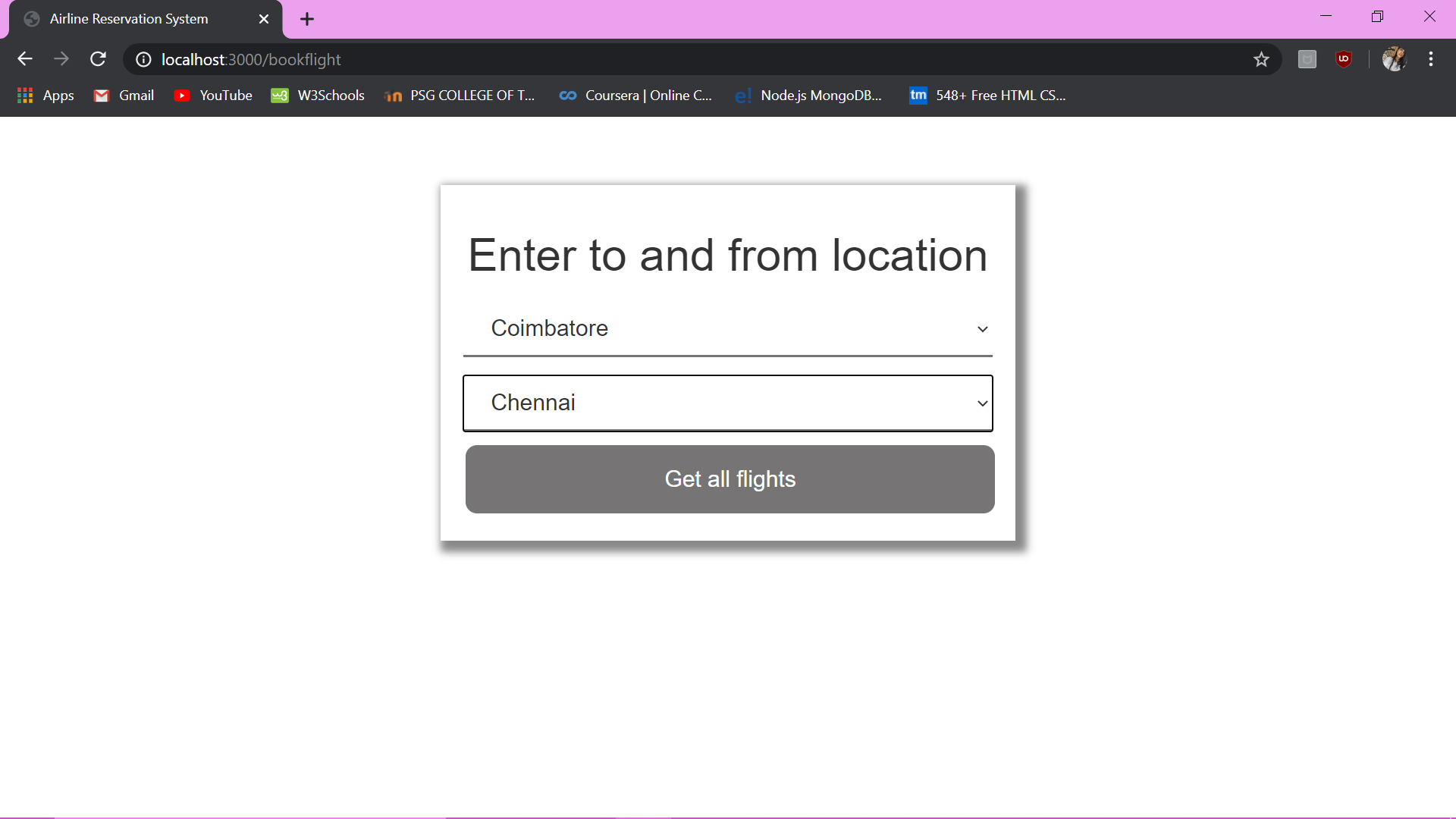


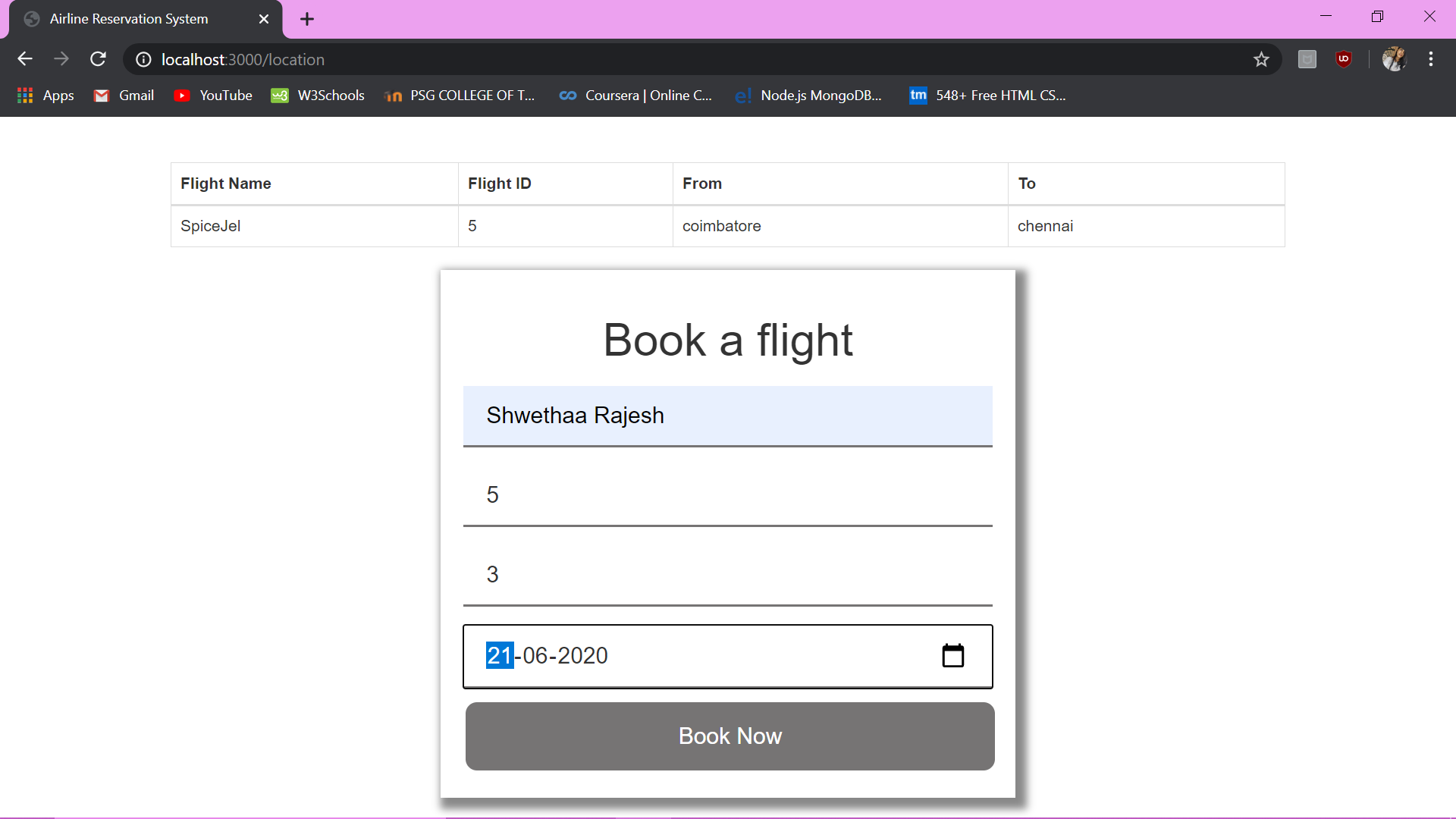


Our second page with all the options

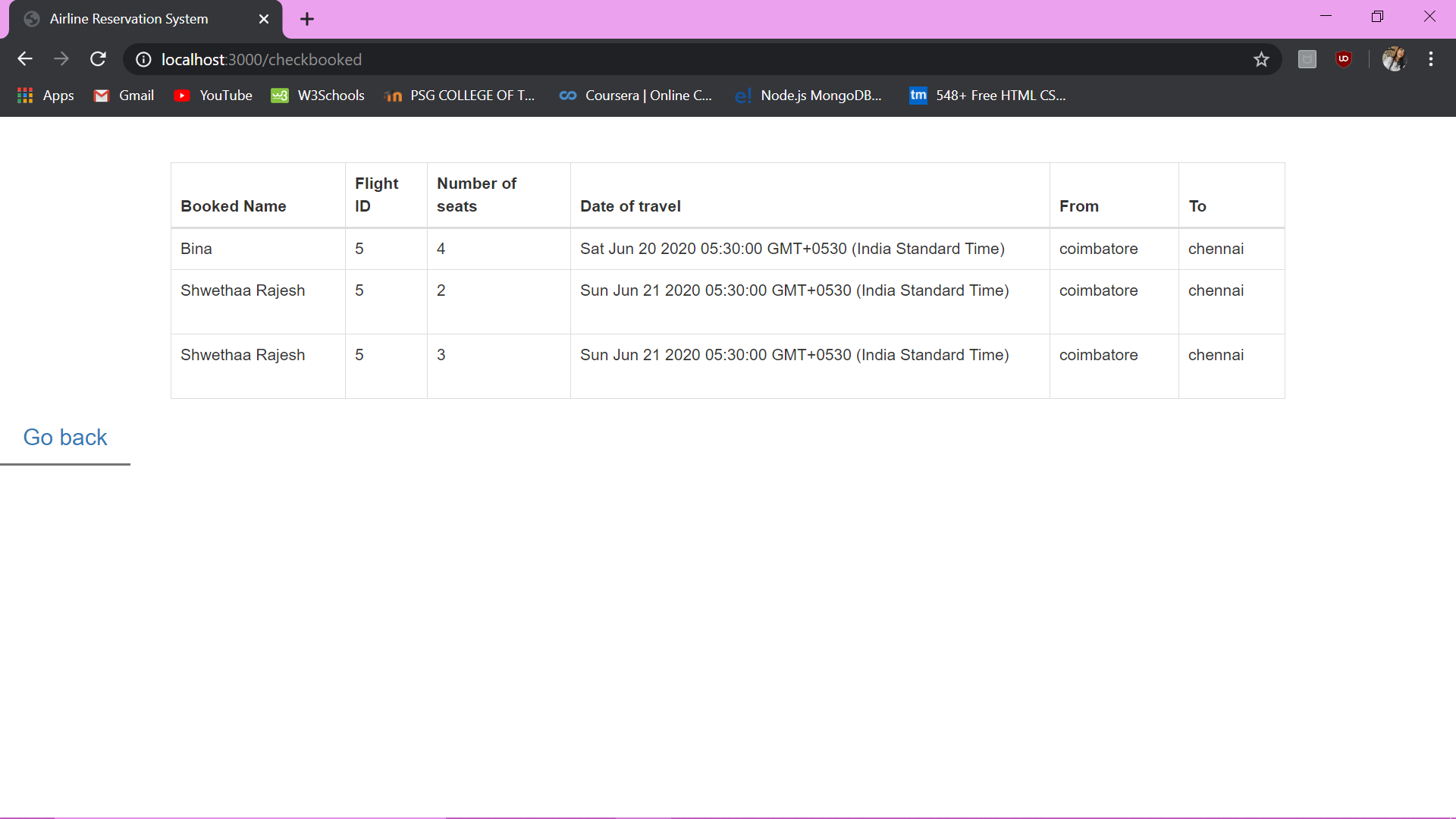


Book Flight:

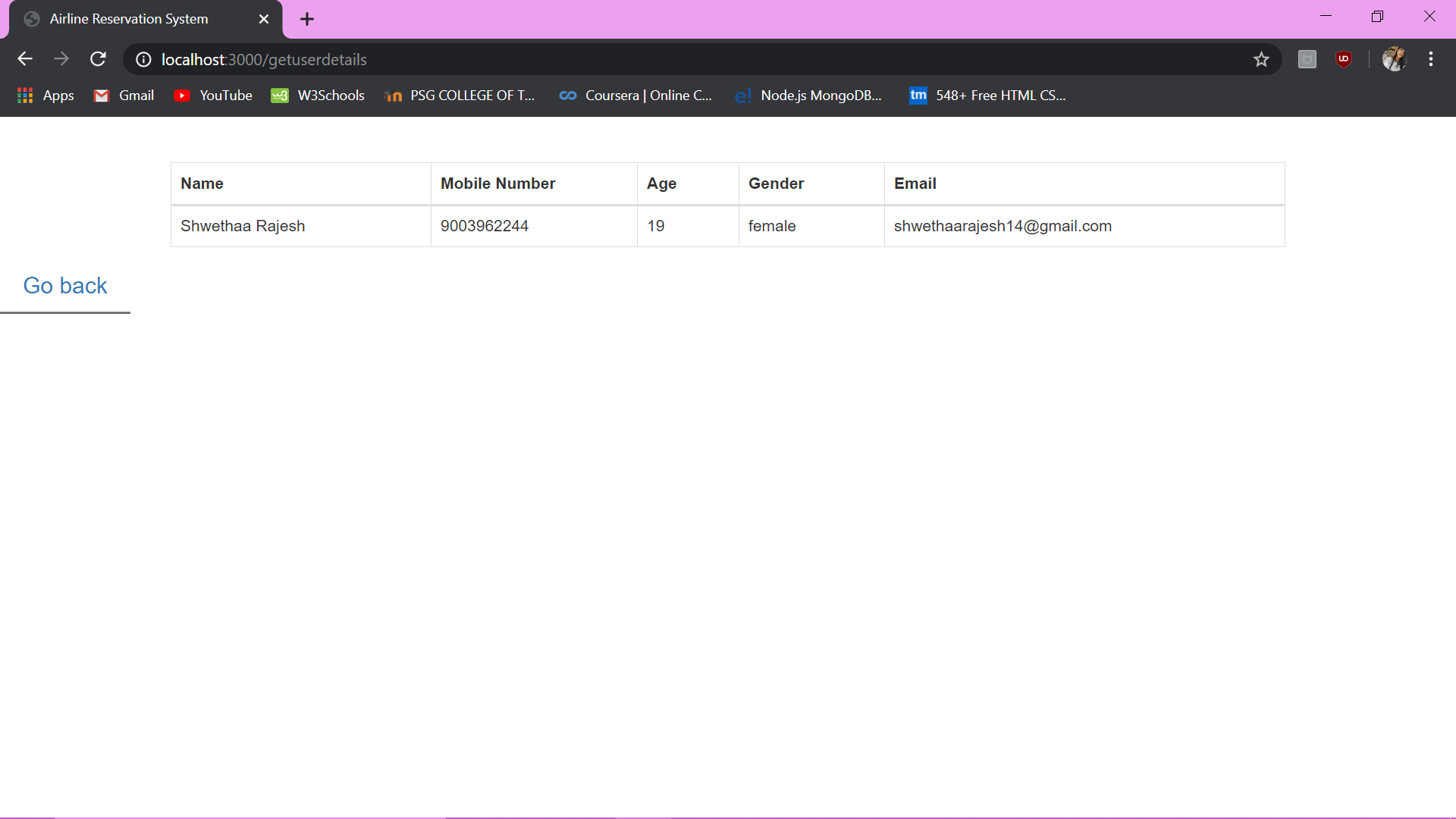




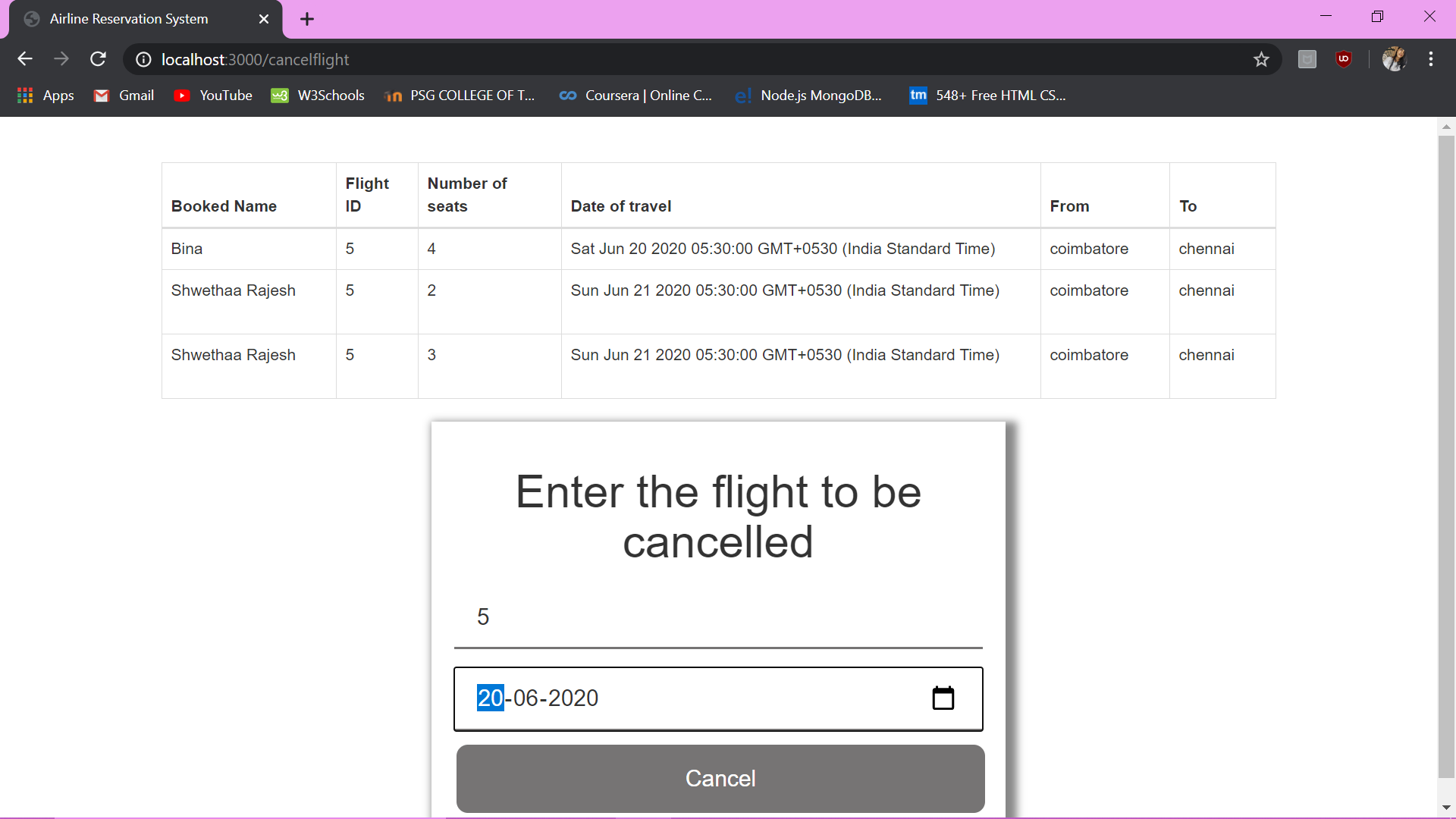
View all bookings :

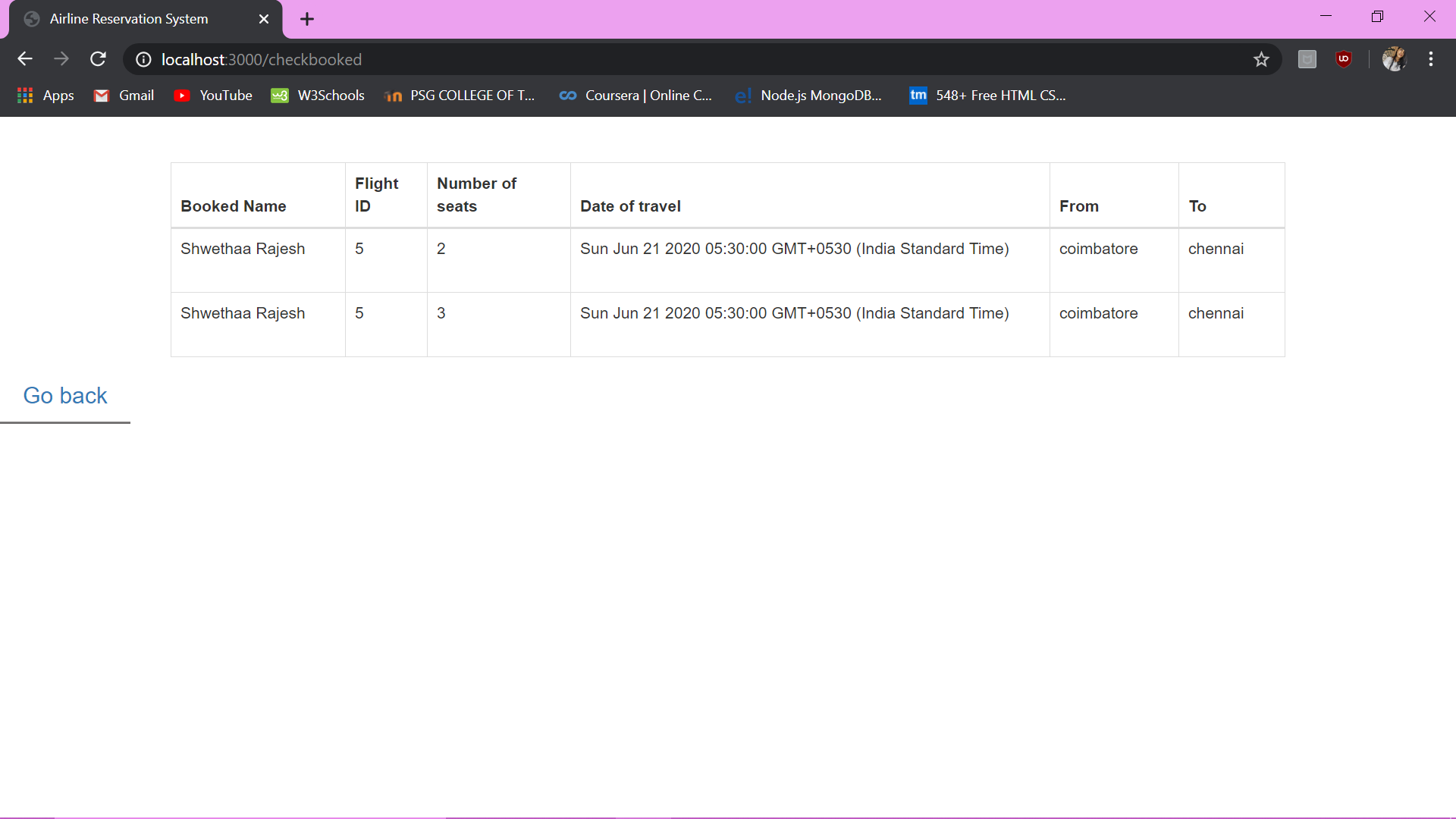


View User Details:

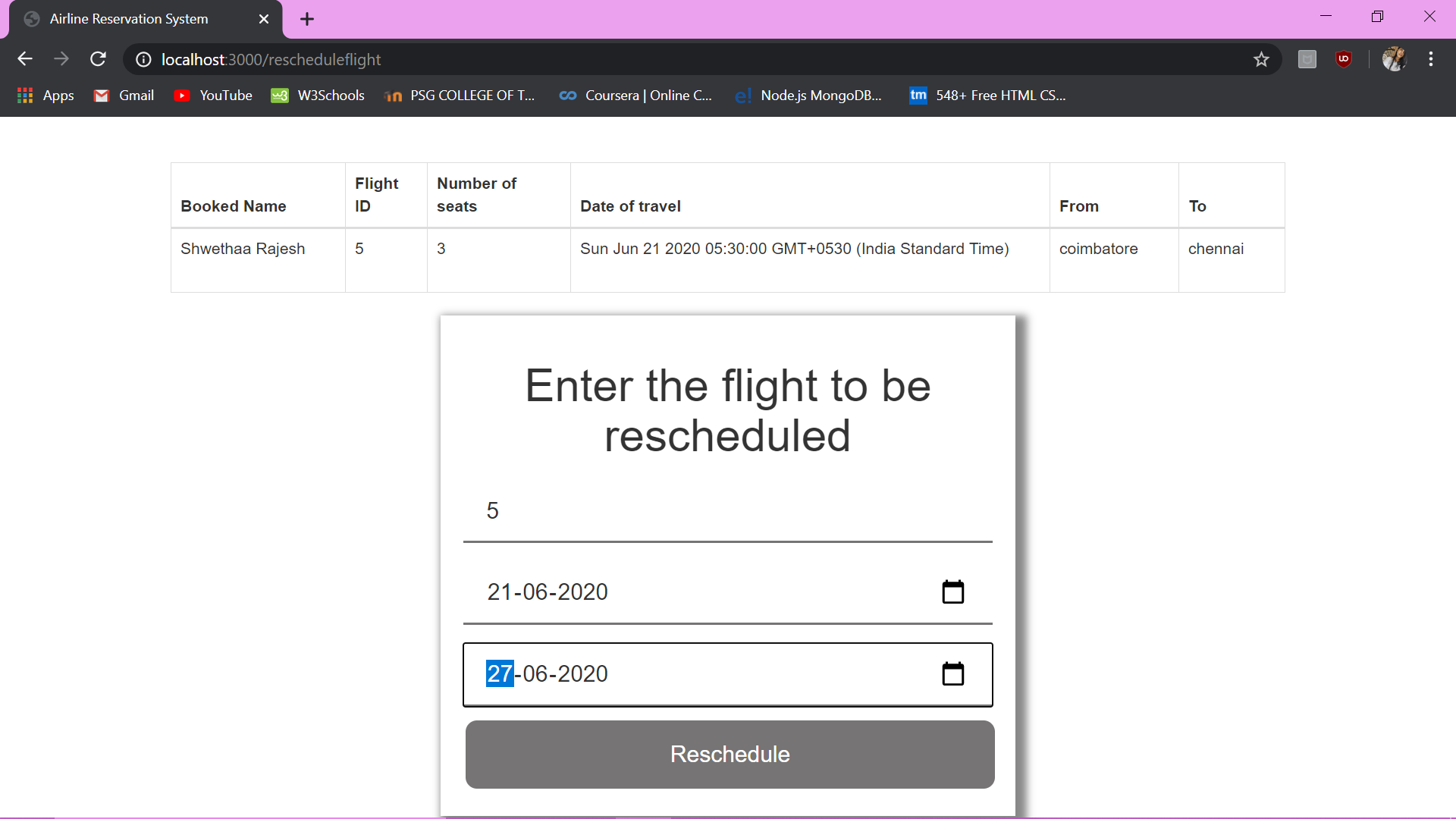


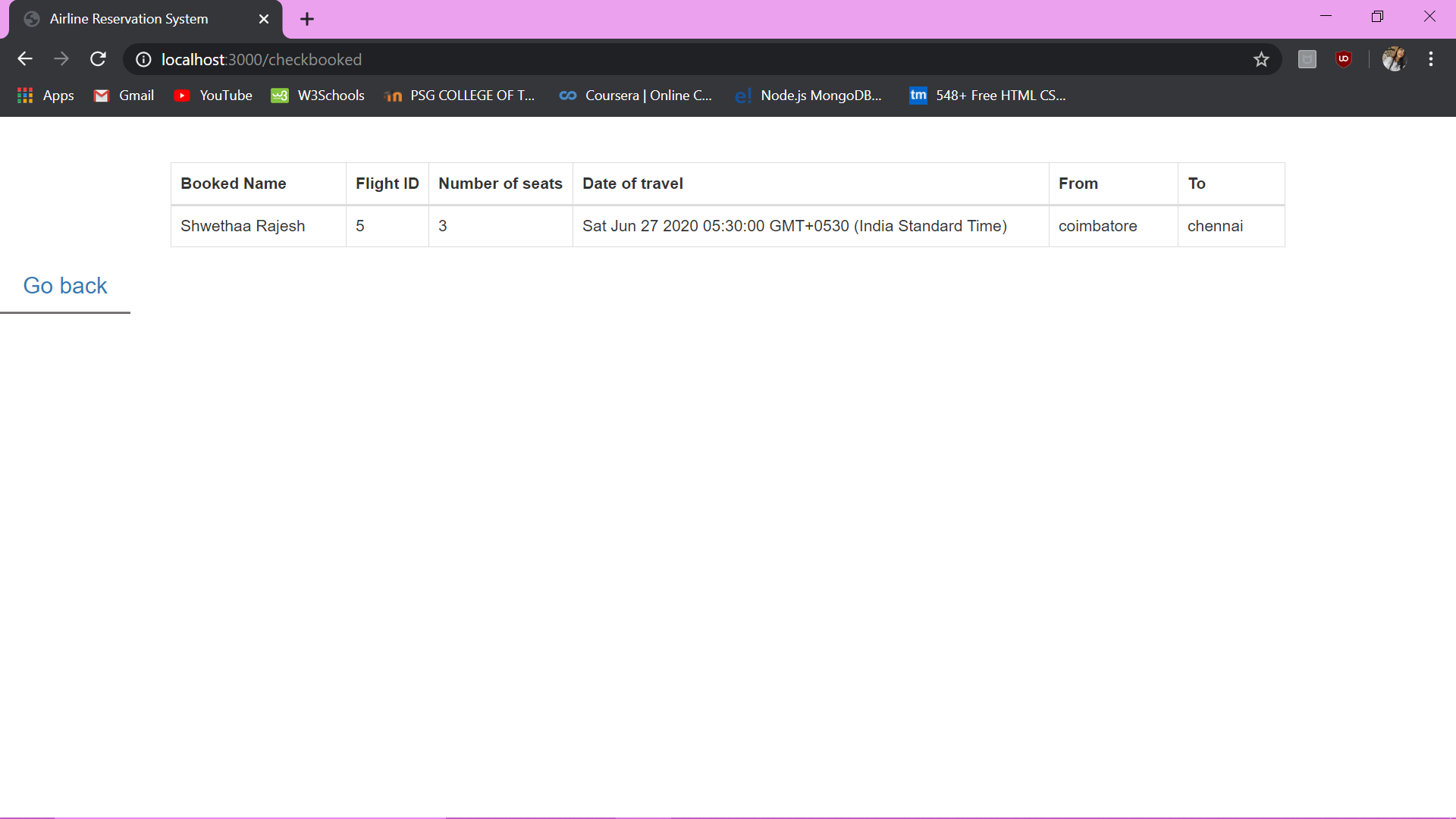
Cancel Flight



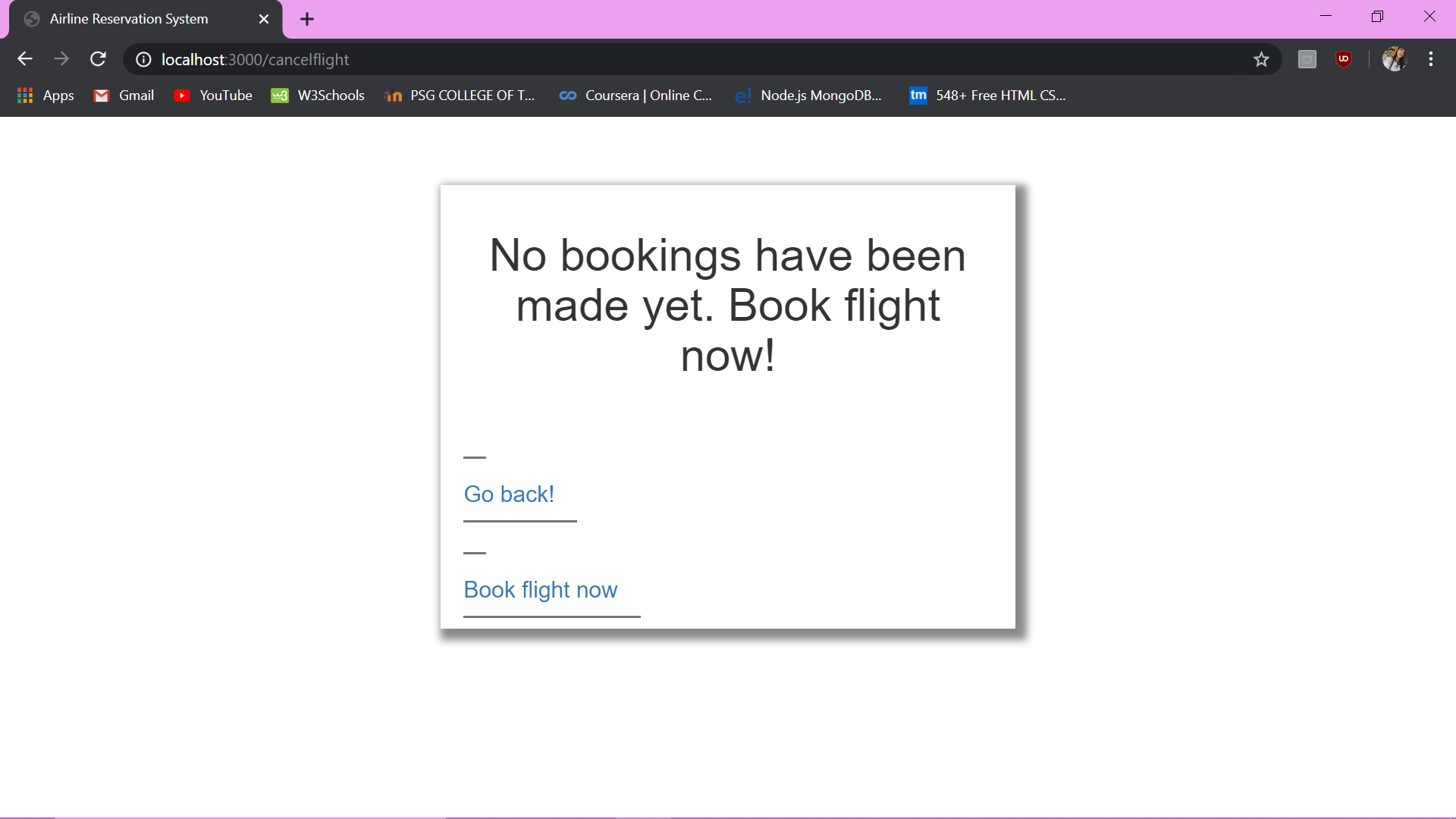


Reschedule Flight:

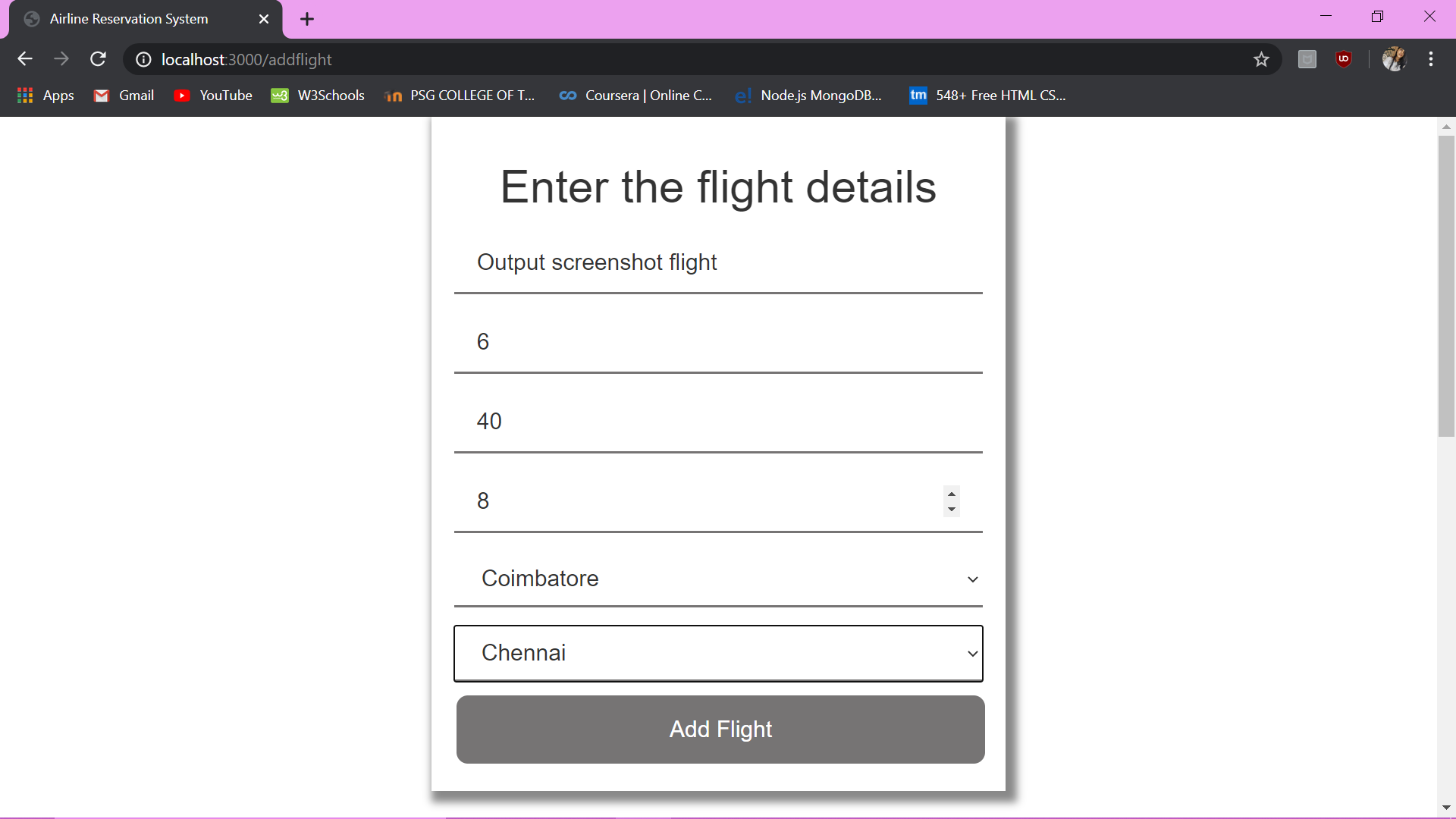


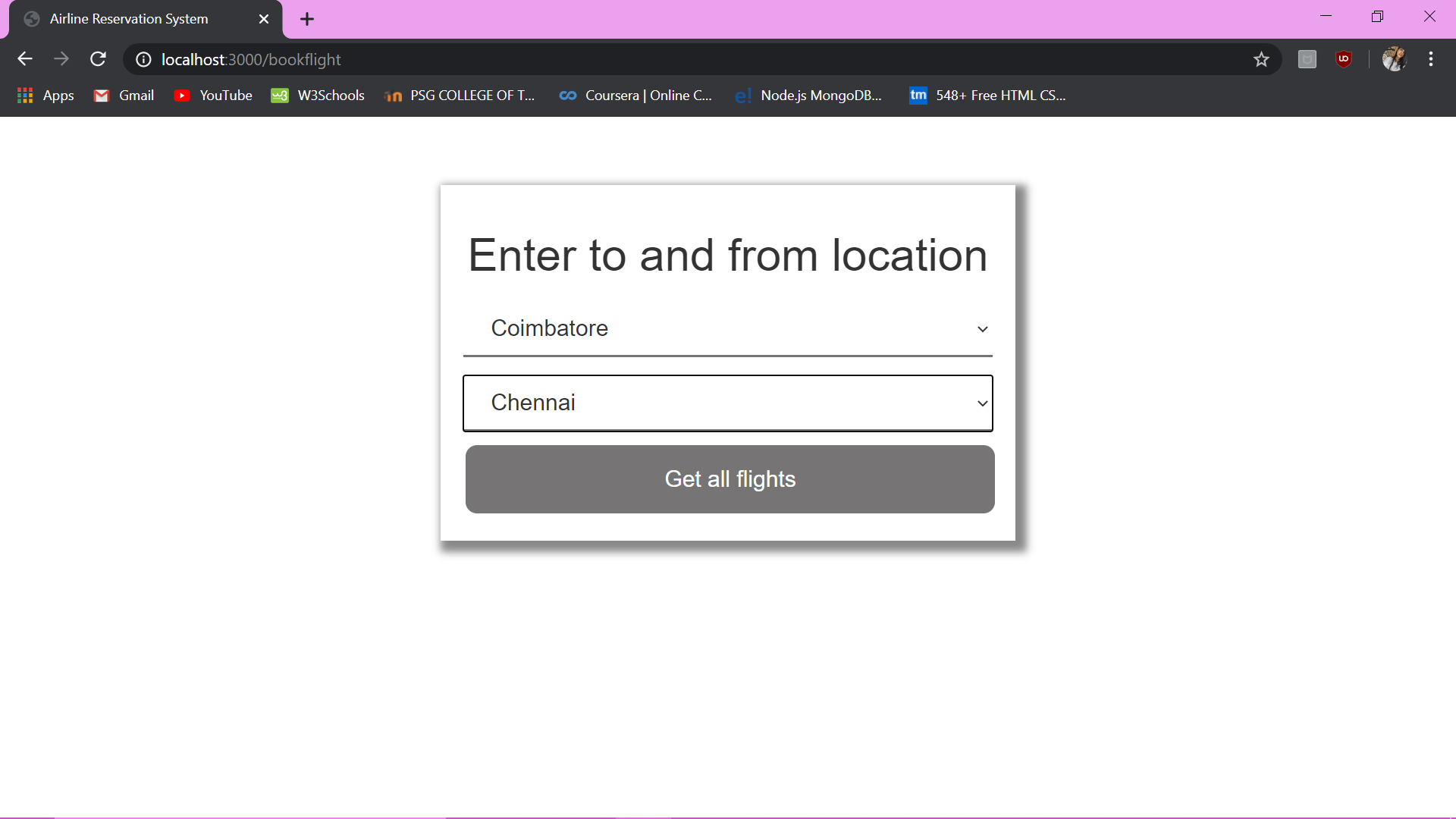


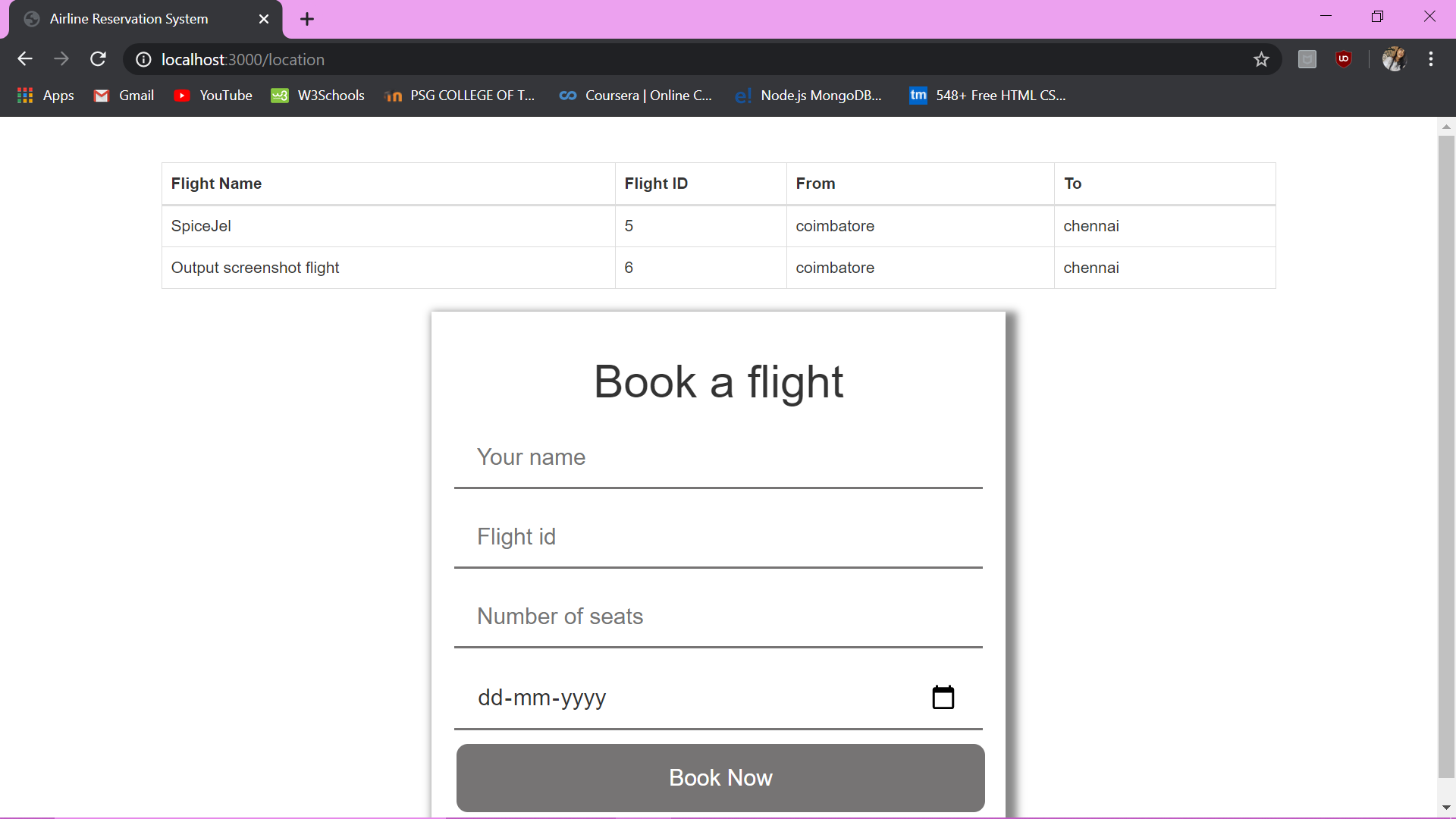
When there are no bookings and trying to cancel/reschedule/ view bookings:



Option to add flight(this feature might either be removed or added along with an admin login)







1. **Conclusion**

Thereby, an airline reservation system is not only easy to implement but also saves us from the hassle of manual labour. It is fast, simple and extremely efficient. We used MongoDB and Node.js. MongoDB is more flexible and easy to use. Node.js is very fast and provides us with modules for better development. Node.js helps communication between the server and the client side.

Thus, an airline reservation system with user login, flight booking, rescheduling and cancelling is successfully implemented

1. **Future Enhancements**

The following future enhancements can be made to our project of an airline reservation system:

1. Add an admin feature where the admin can monitor all flights and add new flights
2. Admin feature to view particular user’s details.
3. Admin feature to view all bookings on a particular day
4. Add an option for users to check in to the flight via the system.
5. Add an option for users to reserve food on the flight
6. Add an option for users to book hotels to the cities they go to.
7. Add more flights with many more cities and countries as well.
8. **References**
9. <https://www.edureka.co/blog/node-js-mongodb-tutorial/>
10. <https://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm>
11. <https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9>
12. <https://www.mongodb.com/blog/post/quick-start-nodejs-mongodb--how-to-get-connected-to-your-database>