Creating a real-time bus tracking system with booking functionality, live bus location tracking using Google Maps API, and a visually appealing React.js front-end is a complex project that would require a significant amount of code and setup. I can provide you with a high-level overview and steps to get you started, but creating the entire project is beyond the scope of a single response. Below are the key steps and guidelines:

\*\*Step 1: Set Up Your Development Environment\*\*

- Ensure you have Node.js and npm installed.

- Install React.js using `npx create-react-app bus-tracking-system`.

- Set up your MySQL database and create tables for buses, users, bookings, and bus schedules.

- Obtain a Google Maps API key for map integration.

\*\*Step 2: Design the Front-End\*\*

- Create a visually appealing user interface using React.js, HTML, CSS, and potentially a UI library like Material-UI or Ant Design.

- Implement transitions and effects using CSS animations or libraries like React Transition Group.

- Organize your project structure with components for booking, tracking, user authentication, and more.

- Set up React Router for navigation between different pages.

\*\*Step 3: Implement User Authentication\*\*

- Use a library like Firebase Authentication or implement custom authentication with JWT for user sign-up and login.

\*\*Step 4: Create Bus Booking Functionality\*\*

- Develop a booking form where users can select a bus, date, and time.

- Implement logic to check bus availability and make bookings.

- Store booking data in the MySQL database.

\*\*Step 5: Real-Time Bus Tracking\*\*

- Integrate Google Maps JavaScript API into your React app.

- Display the map and track the live location of the selected bus using GPS data.

- Implement a feature to show upcoming bus stops and estimated arrival times.

\*\*Step 6: Fetch Data from MySQL\*\*

- Use Axios or the fetch API to communicate with your MySQL database from your React app.

- Retrieve bus schedules, booking information, and user data as needed.

\*\*Step 7: Implement Google Maps Integration\*\*

- Use the Google Maps API to display the bus's location on the map.

- Update the bus's location in real-time.

\*\*Step 8: Styling and Effects\*\*

- Continue refining the user interface with styling and animations.

- Ensure a visually appealing color scheme and layout.

\*\*Step 9: Testing and Debugging\*\*

- Test your application thoroughly to ensure all features work as expected.

- Debug any issues that arise.

\*\*Step 10: Deployment\*\*

- Build your React app using `npm run build`.

- Deploy your app on a hosting platform like Netlify, Vercel, or AWS.

- Set up environment variables for sensitive information like API keys.

Here's a simplified example of a React component to get you started with integrating Google Maps. Make sure to install the `@react-google-maps/api` package to use it.

```jsx

import React from 'react';

import { GoogleMap, LoadScript, Marker } from '@react-google-maps/api';

const MapContainer = ({ busLocation }) => {

const mapStyles = {

height: '400px',

width: '100%',

};

return (

<LoadScript googleMapsApiKey="YOUR\_GOOGLE\_MAPS\_API\_KEY">

<GoogleMap

mapContainerStyle={mapStyles}

zoom={15}

center={busLocation}

>

<Marker position={busLocation} />

</GoogleMap>

</LoadScript>

);

};

export default MapContainer;

```

Remember to replace `"YOUR\_GOOGLE\_MAPS\_API\_KEY"` with your actual Google Maps API key.

This is just a starting point, and building a complete bus tracking system with all the features you mentioned will require a lot of code and effort. You may also need additional libraries and frameworks for handling real-time data updates, managing state, and more. It's recommended to break down the project into smaller tasks and work on them one by one while referring to documentation and tutorials as needed.