

Lab #14(a)

Lab#14(a) Implement Insert Operation to add new record of student in Student Information System

Final Outcome:



The screenshot displays a web application interface for a Student Management System. At the top, a dark blue header bar contains a logo on the left, the text "STUDENT MANAGEMENT SYSTEM" in the center, and "STUDENTS" on the right. Below the header, a white form is presented with the following fields: a text input for "albababab", a text input for "xbxydyd", a dropdown menu for "Male", a date input for "12-03-2002", a text input for "kzy", a text input for "abc", a text input for "czzvzv", a text input for "098765", a text input for "abc@gmail.com", a text input for "234567890", a text input for "qabc", and a text input for "99791255". At the bottom of the form is a "Submit" button.

Code Snippet:

```
// Route handler to create a new student
exports.createStudent = (req, res) => {
  // Extract student data from req.body and insert it into the SQLite database
  const {
    first_name,
    last_name,
    gender,
    date_of_birth,
    address,
    city,
    state,
    zipcode,
    ,
    email,
    phone_number,
    guardian_name,
    guardian_phone,
  } = req.body;

  const stmt = db.prepare(
    "INSERT INTO students (first_name, last_name, gender, date_of_birth, address, city, state, zipcode, email, phone_number, guardian_name, guardian_phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
  );

  stmt.run(
    [
      first_name,
      last_name,
      gender,
      date_of_birth,
      address,
      city, state,
      zipcode, email,
      phone_number,
      guardian_name,
      guardian_phone,
    ],
    (err) => {
      if (err) {
```

Lab #14(a)

```
console.error(err);
res.status(500).send("Internal Server Error");
return;

res.redirect("/sims/all");
}
);

stmt.finalize();
};

//delete student from database exports.deleteStudent = (req, res) => { const id = req.params.id; const
stmt = db.prepare("DELETE FROM students WHERE id = ?");
stmt.run([id], (err) => {
if (err) {
console.error(err);
res.status(500).send("Internal Server Error"); return;
}

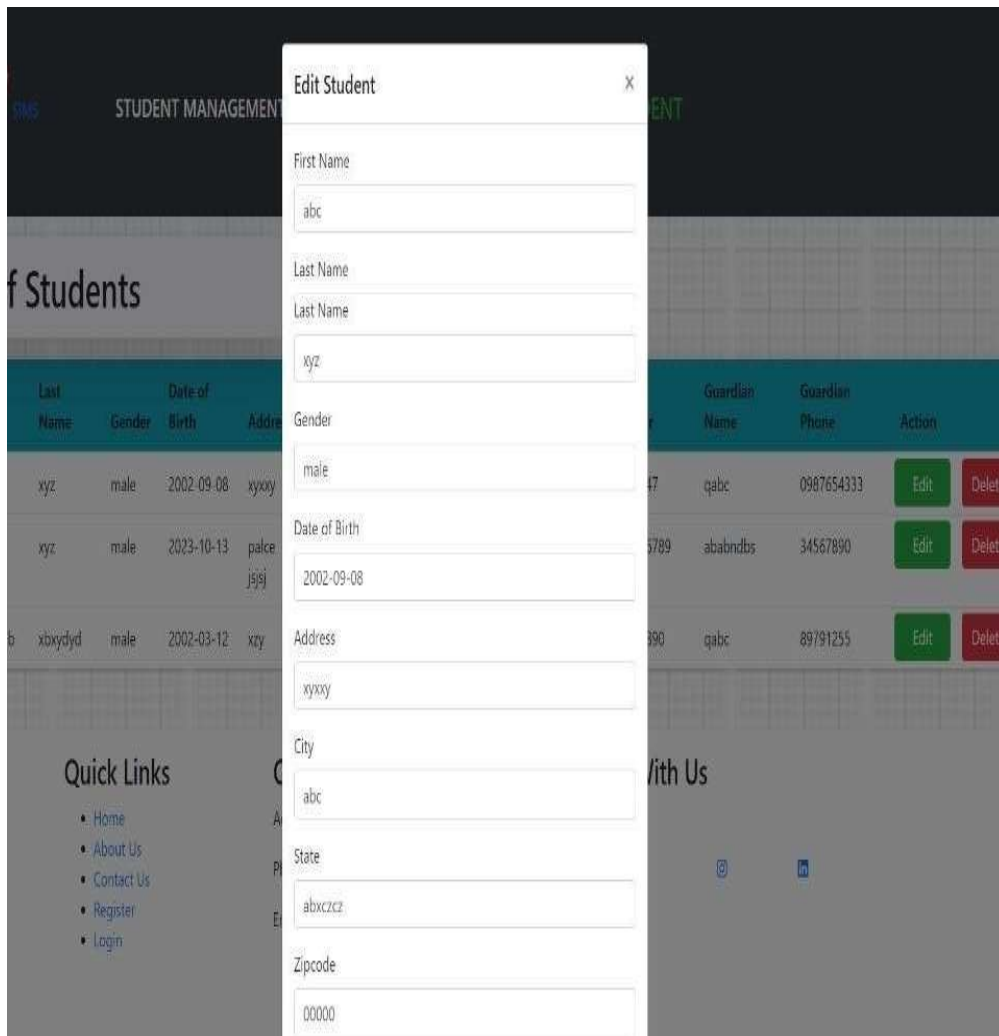
res.redirect("/sims/all");
});

stmt.finalize();
};
```

Lab #14(b)

Lab#14(b) Implement Update Operation to modify any record of student in Student Information System

Final Outcome:



The screenshot displays a web application for student management. A modal window titled "Edit Student" is open, allowing for the modification of a student's information. The modal contains the following fields:

- First Name: abc
- Last Name: xyz
- Gender: male
- Date of Birth: 2002-09-08
- Address: xxxxy
- City: abc
- State: abxczcz
- Zipcode: 00000

The background interface includes a "STUDENT MANAGEMENT" header, a table of students, and a "Quick Links" sidebar with links to Home, About Us, Contact Us, Register, and Login.

Code Snippet:

```
// Route handler to update a student by ID
exports.updateStudent = (req, res) => {
  const studentId = req.params.id;

  const {
```

Lab #14(b)

```
first_name,  
last_name,  
gender,  
date_of_birth,  
address,  
city,  
state,  
zipcode,  
email,  
phone_number,  
guardian_name,  
guardian_phone,  
} = req.body;
```

```
// Begin a transaction db.run("BEGIN  
TRANSACTION", function (beginErr) { if (beginErr)  
{  
  console.error(beginErr);  
  res.status(500).send("Internal Server Error");  
  return;  
}
```

```
const stmt = db.prepare(  
  "UPDATE students SET first_name = ?, last_name = ?, gender = ?, date_of_birth = ?, address = ?, city  
= ?, state = ?, zipcode = ?, email = ?, phone_number = ?, guardian_name = ?, guardian_phone = ? WHERE  
id = ?"  
);
```

```
stmt.run(  
  first_name,  
  last_name,  
  gender,  
  date_of_birth,  
  address,  
  city,  
  state,  
  zipcode  
  ,  
  email,  
  phone_number,  
  guardian_name,  
  guardian_phone,  
  studentId,  
  (err) => { if (err) {
```

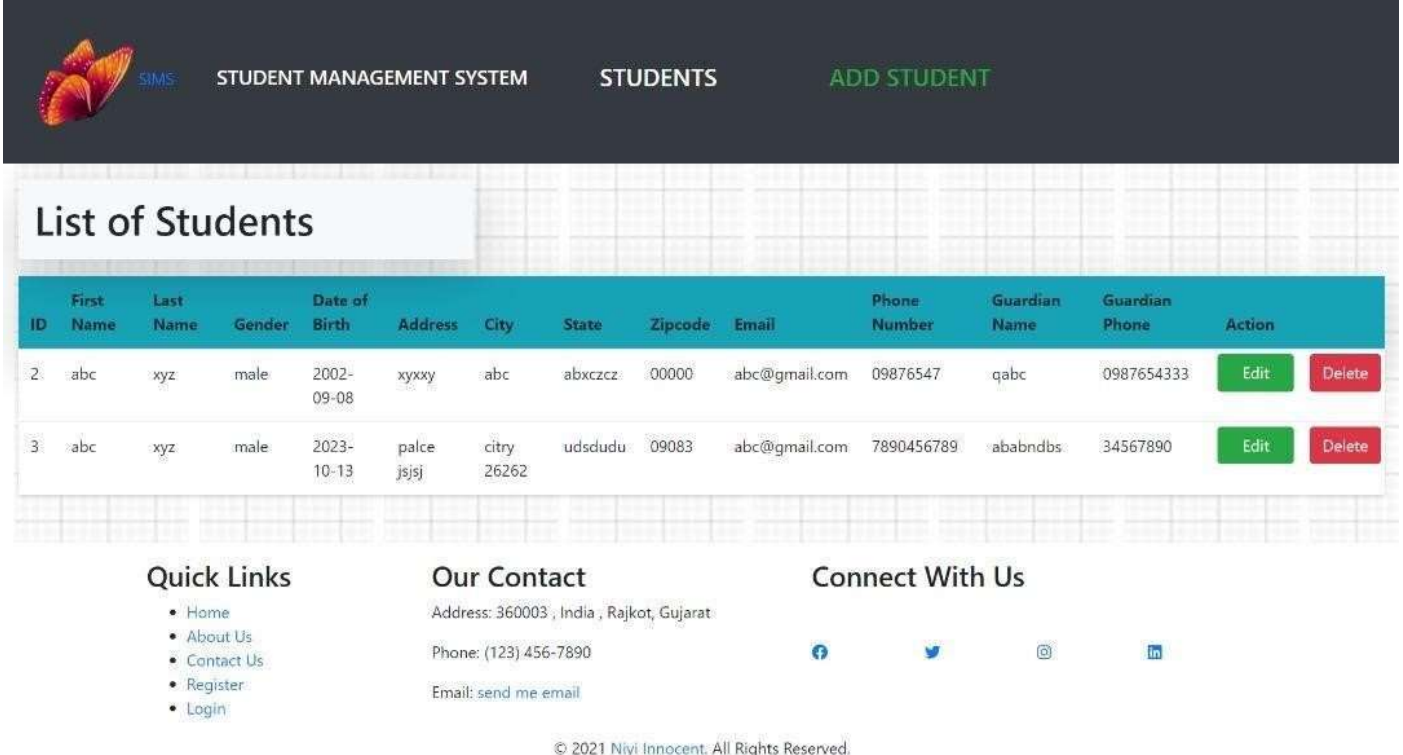
Lab #14(b)

```
console.error(err);
// Rollback the
transaction in case
of an error
db.run("ROLLBACK
", (rollbackErr) => {
if (rollbackErr) {
console.error(rollba
ckErr);
}
res.status(500).sen
d("Internal Server
Error");
});
} else {
// Commit the
transaction
db.run("COMMIT",
(commitErr) => {
if (commitErr) {
console.error(com
mitErr);
res.status(500).sen
d("Internal Server
Error");
} else {
console.log("Studen
t updated
successfully.");
res.redirect("/sims
/all");
}
});
}
}
});

stmt.finalize();
});
};
```

Lab#14(c). Implement Delete Operation to remove any student from Student Information System.

Final Outcome:



The screenshot displays the SIMS (Student Information Management System) interface. At the top, there's a navigation bar with a logo, the text "STUDENT MANAGEMENT SYSTEM", and buttons for "STUDENTS" and "ADD STUDENT". Below this, a "List of Students" section shows a table with student details. The table has columns for ID, First Name, Last Name, Gender, Date of Birth, Address, City, State, Zipcode, Email, Phone Number, Guardian Name, Guardian Phone, and Action. Two students are listed: one with ID 2 and another with ID 3. Each student row has "Edit" and "Delete" buttons. At the bottom, there are sections for "Quick Links" (Home, About Us, Contact Us, Register, Login), "Our Contact" (Address, Phone, Email), and "Connect With Us" (social media icons). A copyright notice at the bottom reads "© 2021 Niyi Innocent. All Rights Reserved."

ID	First Name	Last Name	Gender	Date of Birth	Address	City	State	Zipcode	Email	Phone Number	Guardian Name	Guardian Phone	Action
2	abc	xyz	male	2002-09-08	xyxy	abc	abxczcz	00000	abc@gmail.com	09876547	qabc	0987654333	Edit Delete
3	abc	xyz	male	2023-10-13	palce jsjsj	city 26262	udsdu	09083	abc@gmail.com	7890456789	ababndbs	34567890	Edit Delete

Code Snippet:

```
//delete student from database exports.deleteStudent = (req,
res) => { const id = req.params.id; const stmt =
db.prepare("DELETE FROM students WHERE id = ?");

stmt.run([id], (err) => {
  if (err) {
    console.error(err);
    res.status(500).send("Internal Server Error");
    return;
  }

  res.redirect("/sims/all");
});
```



```
stmt.finalize();  
};
```