Express Practice App

Npm init
npm I express

Postman: 127.0.0.1:3000

App.js

```
const express = require('express');

const app = express();

app.get('/', (req, res) => {
    res.status(200).send('Hello from the server side!');
    //output: Hello from the server side!

    //or
    res.status(200).json({ message: "Hello from the server side!", app: 'MyExpress'
});
    output: {
    "message": "Hello from the server side!",
    "app": "MyExpress"
}

});
const port = 3000;
```

API with get request

```
const express = require("express");
const fs = require('fs')

const app = express();

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);

app.get("/api/v1/tours", (req, res) => {
  res.status(200).json({
      status: "success",
      results: tours.length,
    data: {
      tours,
    },
  });
});

const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

Postman: 127.0.0.1:3000/api/v1/tours ……..raw, json

Postman thaka Jason dissi:

```
{
    "name": "Test Tours",
     "duration": 5,
    "difficulty": "easy"
}
```

And

Terminal a amk javascript object disse:

```
{name: 'Test Tours', duration: 5, difficulty: 'easy' }
```

Post request:

```
const express = require("express");
const fs = require('fs')

const app = express();

app.use(express.json());//middleware

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);

app.get("/api/v1/tours", (req, res) => {
  res.status(200).json({
      status: "success",
      results: tours.length,
    data: {
      tours,
    },
  });
});

app.post("/api/v1/tours", (req, res) => {
    console.log(req.body);
    res.send('Done')
});
//post request a data client theke server a post kori. So ai data req er modda pai. Akhon kotha holo express does not put the body data on the request. SO ai jonno middleware use korbo
const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});

//127.0.0.1:3000/api/v1/tours
```

In post write the data in Rest API code:

```
app.post("/api/v1/tours", (req, res) => {
    const newId = tours[tours.length - 1].id + 1;
    const newTour = Object.assign({ id: newId }, req.body);

    tours.push(newTour);
    fs.writeFile(`${__dirname}/dev-data/data/tours-simple.json`,
JSON.stringify(tours), err => {
        res.status(201).json({//201 name datta create hoise
            status: 'success',
            data: {
                tour: newTour
            }
        })
    });
});
```

Get req with parameter:

```
app.get("/api/v1/tours/:id", (req, res) => {
  console.log(req.params);
  res.status(200).json({
    status: "success",
  });
});
```

Postman: 127.0.0.1:3000/api/v1/tours/5

Terminal ans: { id: '5' }

```
app.get("/api/v1/tours/:id/:x/:y", (req, res) => {
  console.log(req.params);
  res.status(200).json({
    status: "success",
  });
});
```

>"/api/v1/tours/:id/:x/:y"

Postman: 127.0.0.1:3000/api/v1/tours/5/23/25 pura ta na dile error pabo

Terminal: { id: '5', x: '23', y: '25' }

>"/api/v1/tours/:id/:x/:y?" Question mark ase mane optional

Postman: 127.0.0.1:3000/api/v1/tours/5/23

Terminal: { id: '5', x: '23', y: undefined }

Const id = req.params.id * 1;

Javascript a string jeta number er moto dekte take number er shate muntiply korly number kora jai

//Find tour

```
//find tour
app.get("/api/v1/tours/:id", (req, res) => {
  console.log(req.params);
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
});
```

Patch request mane update the request:

```
//handling Patch request/ update tour
app.patch("/api/v1/tours/:id", (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
});
```

Postman: 127.0.0.1:3000/api/v1/tours/8

{
"duration": 13
}
Output: {

```
    "status": "success",
    "data": {
        "tour": "Update tour here.."
    }
}
```


```
//delete request
app.delete("/api/v1/tours/:id", (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null
  });
});
```

Postman: 127.0.0.1:3000/api/v1/tours/9

//refectore code using route 1

```
const express = require("express");
const fs = require("fs");

const app = express();

app.use(express.json());

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);

const getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    results: tours.length,
    data: {
      tours,
    },
  });
};

const getTour = () => (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
```

```
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};

//New object create korte gele amder id niya chinta korte hobe na database a she
automatically ID create hobe or paya jabe.
const createTour = () => (req, res) => {
  const newId = tours[tours.length - 1].id + 1;  //tours.length-1 mane last er tour
& then 1 add tar shate.
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        status: "success",
        data: {
          tour: newTour,
        },
      });
    }
  );
};

const updateTour = () => (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};
```

```
const deleteTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null,
  });
};


//get api
app.get("/api/v1/tours", getAllTours);

//crate tour in API
app.post("/api/v1/tours", createTour);

//find tour
app.get("/api/v1/tours/:id", getTour);

//handling Patch request/ update tour
app.patch("/api/v1/tours/:id", updateTour);

1.//delete request
2.app.delete("/api/v1/tours/:id", deleteTour);

3.const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});

//127.0.0.1:3000/api/v1/tours
```

//refectore code 2

```
//refectore using route
app.route("/api/v1/tours").get(getAllTours).post(createTour);

app.route("/api/v1/tours/:id").patch(updateTour).delete(deleteTour);
```

Uporer code gula :

```
const express = require("express");
const fs = require("fs");

const app = express();

app.use(express.json()); //middleware

const tours = JSON.parse(
  //server start er shate shate e tours er data asisha pore tai amder restart korte hoi server pry shomoi
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);
//get api
app.get("/api/v1/tours", (req, res) => {
  res.status(200).json({
    status: "success",
    results: tours.length,
    data: {
      tours,
    },
  });
});

//find tour
app.get("/api/v1/tours/:id", (req, res) => {
  console.log(req.params);
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
});

//post request
app.get("/api/v1/tours", (req, res) => {
  res.status(200).json({
```

```
    status: "success",
    results: tours.length,
    data: {
      tours,
    },
  });
});

//crate tour in API
app.post("/api/v1/tours", (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        //201 name datta create hoise
        status: "success",
        data: {
          tour: newTour,
        },
      });
    }
  );
});

//handling Patch request/ update tour
app.patch("/api/v1/tours/:id", (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
});

//delete request
app.delete("/api/v1/tours/:id", (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
```

```
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null,
  });
});

const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});

//127.0.0.1:3000/api/v1/tours
```

Hulf code Here :

```
const express = require("express");
const fs = require("fs");

const app = express();

app.use(express.json());
```

```
const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);

const getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    results: tours.length,
    data: {
      tours,
    },
  });
};

const getTour = () => (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};

const createTour = () => (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        status: "success",
        data: {
          tour: newTour,
        },
      });
    }
  );
};
```

```
const updateTour = () => (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};

const deleteTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null,
  });
};



//get api
app.get("/api/v1/tours", getAllTours);

//crate tour in API
app.post("/api/v1/tours", createTour);

//find tour
app.get("/api/v1/tours/:id", getTour);

//handling Patch request/ update tour
app.patch("/api/v1/tours/:id", updateTour);

//delete request
app.delete("/api/v1/tours/:id", deleteTour);

//refectore using route
app.route("/api/v1/tours").get(getAllTours).post(createTour);
```

```
app.route("/api/v1/tours/:id").patch(updateTour).delete(deleteTour);

const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});

//127.0.0.1:3000/api/v1/tours




//middleware and the req, res cycle
//it's called middleware because middleware is function that is executed between. She modda bosha receive kore request and send kore response. In express everything is middleware.

//je middleware gula app a use kori shai Shob middleware k ek shate bola hoi middleware stack.


//create our own middleware
const express = require("express");
const fs = require("fs");
const morgan = require("morgan");
const app = express();
// app.use(morgan("dev"));
app.use(express.json());

//own middleware
app.use((req, res, next) => {
  console.log('Hello from the middleware');
  next();
})

app.use((req, res, next) => {
  req.requestTime = new Date().toISOString();
  next()
})

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);
//route handler
const getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    requestedAt: req.requestTime,
```

```
    results: tours.length,
    data: {
      tours,
    },
  });
};
//get route
app.get("/api/v1/tours", getAllTours);

//Now after global middleware then route
const port = 3000;

app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});



//3rd party Middleware
// morgan popular loging midleware. its a regular dependiencie not developer
dependiencie that why we not use -save dev

const express = require("express");
const morgan = require('morgan');

const app = express();

app.use(express.json());
app.use(morgan('dev'));
//shotik url dile terminal a answer pabo  GET /api/v1/tours 200 63.000 ms - 8886
//vul url dile terminal pabo GET /api/v1/tourserytyuyu 404 5.332 ms - 159



//implement user route
const express = require("express");
const app = express();
app.use(express.json());

const getAllUser = (req, res) => {
  res.status(500).json({
    status: 'error',
    message: 'This route is not yet defined!'
  })
}
const getUser = (req, res) => {
  res.status(500).json({
    status: "error",
```

```
    message: "This route is not yet defined!",
  });
};
const createUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const updateUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const deleteUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};

app.route("/api/v1/users").get(getAllUser).post(createUser);

app
  .route("/api/v1/users/:id")
  .get(getUser)
  .patch(updateUser)
  .delete(deleteUser);

const port = 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

creating and Mounting Multiple Router:

```
const express = require("express");
const fs = require("fs");

const app = express();
```

```
app.use(express.json());

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/dev-data/data/tours-simple.json`)
);

const getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    requestedAt: req.requestTime,
    results: tours.length,
    data: {
      tours,
    },
  });
};


const getTour = (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};


const createTour = (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        status: "success",
        data: {
          tour: newTour,
        },
```

```
      });
    }
  );
};

const updateTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};

const deleteTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null,
  });
};


//user
const getAllUsers = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const getUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const createUser = (req, res) => {
  res.status(500).json({
```

```
    status: "error",
    message: "This route is not yet defined!",
  });
};
const updateUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const deleteUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};



//creating and Mounting Multiple Router
//Routes
//mount a new router on a route

const tourRouter = express.Router();
const userRouter = express.Router();


tourRouter.route("/").get(getAllTours).post(createTour);

tourRouter.route('/:id').get(getTour).patch(updateTour).delete(deleteTour);

//user Router
userRouter.route("/").get(getAllUsers).post(createUser);

userRouter.route('/:id').get(getUser).patch(updateUser).delete(deleteUser);

app.use(('/api/v1/tours'), tourRouter);
app.use("api/v1/users", userRouter);

const port = 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

//A better file structure

//refectore code

//update 1

App.js

```
const express = require("express");

const tourRouter = require("./routes/tourRoutes");
const userRouter = require("./routes/userRoutes");

const app = express();

app.use(express.json());

app.use("/api/v1/tours", tourRouter);
app.use("api/v1/users", userRouter);

const port = 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

tourRoutes.js

```
const express = require("express");
const fs = require("fs");

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/../dev-data/data/tours-simple.json`)
);

const getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    requestedAt: req.requestTime,
    results: tours.length,
    data: {
      tours,
    },
  });
};

const getTour = (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
```

```
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};

const createTour = (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  console.log(newId);
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        status: "success",
        data: {
          tour: newTour,
        },
      });
    }
  );
};

const updateTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};

const deleteTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
```

```
  res.status(204).json({
    status: "success",
    data: null,
  });
};



const router = express.Router();

router.route("/").get(getAllTours).post(createTour);
router.route("/:id").get(getTour).patch(updateTour).delete(deleteTour);

module.exports = router;
```

userRoute.js

```
const express = require("express");

const getAllUsers = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const getUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const createUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const updateUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
const deleteUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};


const router = express.Router();
```

```
router.route("/").get(getAllUsers).post(createUser);

router.route("/:id").get(getUser).patch(updateUser).delete(deleteUser);


module.exports = router;
```

//update 2

Route

tourRouter.js

```
const express = require("express");
const tourController = require('./../controllers/tourController');


const router = express.Router();

router.route("/").get(tourController.getAllTours).post(tourController.createTour);
router.route("/:id").get(tourController.getTour).patch(tourController.updateTour).de
lete(tourController.deleteTour);

module.exports = router;
```

userRoute.js

```
const express = require("express");
const userController = require('./../controllers/userController');

const router = express.Router();

router.route("/").get(userController.getAllUsers).post(userController.createUser);

router
  .route("/:id")
  .get(userController.getUser)
  .patch(userController.updateUser)
  .delete(userController.deleteUser);


module.exports = router;
```

Controller

tourController.js

```
const fs = require("fs");

const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/../dev-data/data/tours-simple.json`)
);

exports.getAllTours = (req, res) => {
```

```
  res.status(200).json({
    status: "success",
    requestedAt: req.requestTime,
    results: tours.length,
    data: {
      tours,
    },
  });
};

exports.getTour = (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);

  // if (id > tours.length) {
  if (!tour) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};

exports.createTour = (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  console.log(newId);
  const newTour = Object.assign({ id: newId }, req.body);

  tours.push(newTour);
  fs.writeFile(
    `${__dirname}/dev-data/data/tours-simple.json`,
    JSON.stringify(tours),
    (err) => {
      res.status(201).json({
        status: "success",
        data: {
          tour: newTour,
        },
      });
    }
  );
};

exports.updateTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
```

```
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};

exports.deleteTour = (req, res) => {
  if (req.params.id * 1 > tours.length) {
    return res.status(404).json({
      status: "Fail",
      message: "Invalid ID",
    });
  }

  res.status(204).json({
    status: "success",
    data: null,
  });
};
```

userController.js

```
exports.getAllUsers = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
exports.getUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
exports.createUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
exports.updateUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
```

```
  });
};
exports.deleteUser = (req, res) => {
  res.status(500).json({
    status: "error",
    message: "This route is not yet defined!",
  });
};
```

Server.js starting file

Nodemon golbaly install korte pari or npm I nodemon –save-dev korte pari

Server.js

```
const app = require('./app');

const port = 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

Server.js run hobe akhon age.

//Param middleware

//toursRouter.js

```
router.param('id', (req, res, next, val) => {
  //here ID is parameter
  console.log(`Tour: ${val}`);
  next(); //next na thakle ai middleware ai khane stack hoiya thakbe
})
```

//tursController.js

```
exports.checkID = (req, res, next, val) => {
    console.log(`Tour: ${val}`);
   if (req.params.id * 1 > tours.length) {
     return res.status(404).json({
       status: "Fail",
       message: "Invalid ID",
     });
   }
  next();
```

```
}
```

Input: 127.0.0.1:3000/api/v1/tours/2 (get Tour)

Output in terminal: Tour: 2

//Now

//full code

tourRouter.js

```
const express = require("express");
const tourController = require('./../controllers/tourController');


const router = express.Router();


router.param('id', tourController.checkID);


router.route("/").get(tourController.getAllTours).post(tourController.createTour);
router.route("/:id").get(tourController.getTour).patch(tourController.updateTour).de
lete(tourController.deleteTour);


module.exports = router;
```

tourController.js

```
const fs = require("fs");


const tours = JSON.parse(
  fs.readFileSync(`${__dirname}/../dev-data/data/tours-simple.json`)
);


exports.checkID = (req, res, next, val) => {
    console.log(`Tour: ${val}`);
   if (req.params.id * 1 > tours.length) {
     return res.status(404).json({
       status: "Fail",
       message: "Invalid ID",
     });
   }
  next();
}
```

```
exports.getAllTours = (req, res) => {
  res.status(200).json({
    status: "success",
    requestedAt: req.requestTime,
    results: tours.length,
    data: {
      tours,
    },
  });
};

exports.getTour = (req, res) => {
  const id = req.params.id * 1;
  const tour = tours.find((el) => el.id === id);
  res.status(200).json({
    status: "success",
    data: {
      tour,
    },
  });
};

exports.createTour = (req, res) => {
  const newId = tours[tours.length - 1].id + 1;
  console.log(newId);
const newTour = Object.assign({ id: newId }, req.body);

tours.push(newTour);
fs.writeFile(
  `${__dirname}/dev-data/data/tours-simple.json`,
  JSON.stringify(tours),
  (err) => {
    res.status(201).json({
      status: "success",
      data: {
        tour: newTour,
```

```
        },
      });
    }
  );
};

exports.updateTour = (req, res) => {

  res.status(200).json({
    status: "success",
    data: {
      tour: "Update tour here..",
    },
  });
};

exports.deleteTour = (req, res) => {

  res.status(204).json({
    status: "success",
    data: null,
  });
};
//ai function gula validition niya concern na ai gular uddasho ektai ja bola hoi kora.. delete function delete korbe, update function update korbe ai tader kaj. id validation upre automatic kore diyasi aita e express er ekta subida/gun
```

//Chaning multiple middleware

//tourController.js

```
//create own middleware function

//check if body contain the name and price property

//If not, send back 400(bad request(mane user name, price chara e request korse tai bad))
```

```
//And it to the post handler stack
exports.checkBody = (req, res, next) => {
  if (!req.body.name || !req.body.price) {
    return res.status(400).json({
      status: 'fail',
      message: 'Missing name or price'
    })
  }
  next();
}
```

tourRoutr.js

```
//chain multiple middleware
router.route("/").get(tourController.getAllTours).post(tourController.checkBody,
tourController.createTour);
```

postman:create new tour: body.raw

```
{
    "name": "asd",
    "duration": 5,
    "price": 123
}
```

Body:

```
{
    "status": "success",
    "data": {
        "tour": {
            "id": 17,
            "name": "asd",
            "duration": 5,
            "price": 123
        }
    }
```

//Serving static files

http://127.0.0.1:3000/public/overview.html //aitar excess amra kono vabe e pabo na karon kono route define kori nai ai url er jonno & kono handler o nai

```
Cannot GET /public/overview.html
```

So amder file system theke kisur excess nite amra built-in Express middleware use korbo.

```
//middleware
app.use(express.json());
app.use(express.static(`${__dirname}/public`));
```

http://127.0.0.1:3000/overview.html //akhon er public lagbe na url a.

Akhon kotha holo keno public url laglo na? karon jokhon amra kono url open kori tokhon route khuje & route na paile publich folder khuje. So amra akhon public bole disi. ……it sets that folder to the root. So akhon public folder amder root.

//Image open korte chile

http://127.0.0.1:3000/img/pin.png

http://127.0.0.1:3000/img/ //aita dile image pabo na karon akhon ja dilam aita to file na.

So amra deklam static file k ki vabe dekhate pari……static file from a folder and not from a route.

//Environment Variables:

Node js, Express app different environment a run korte pare. 2 ta important environment hose development env & production env.

Ai environment er upor er bitti kore e amra different database use korte pari jamon login on off korte pari, debugging on off, diff setting change korte pari.

Express sets the environment.

Sever.js

```
console.log(app.get('env'))   //sets by express
```

output: development App running on port 3000...

So amra akhon development environment a asi. Env Variable are global variable. But nodeJs nije e onek environment sets kore.

```
console.log(process.env);   //core module
```

Process er biton onek variable ase terminal a dekte pabo log korle e. Je gulo node er modda internally ase. precess everywhere available automatically. Process k require kore ana lage nai.

In Express, many package depend on a special variable call node env. Express nije ai variable define kore na amder korte hoi. So amra eta manually korbo terminall use kore.

NODE_ENV=development nodemon server.js // its not working

SET NODE_ENV=development node server.js //Work it rightly**

Amra sensitive data jamon password, username ai gula set kore rakbo environment variable use kore.

Akhon ato variable to r bar bar to terminal a command diya define korbo na ait ekta config file toiri korbo.
(config.env):
SET NODE_ENV=development
PORT:3000
USER = mila
PASSWORD =123456
//aita ekta txt file er moton node js ai file shomporke jane na. Ai jonno amder ekta package Install korte hobe:

npm i dotenv

Akhon dotenv diya config k call korbo…SO amder config file theke variable read korbe & nodeJs environment a save korbe nicher ai line ta…

```
dotenv.config({path: './config.env'})
```

server.js

```
const app = require('./app');
const dotenv = require('dotenv');

dotenv.config({path: './config.env'})

console.log(app.get('env'));
console.log(process.env);

const port = 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

Akhon user use korbo NODE_ENV & PORT variable.

```

//middleware
console.log(process.env.NODE_ENV);
if (process.env.NODE_ENV === "development") {
  app.use(morgan("dev"));
}
//aita akhon shob file e available
```

Aita holo amra different code k run kori depend kore amra kothi asi development or production.

App.js

```
const express = require("express");
const morgan = require("morgan");

const tourRouter = require("./routes/tourRoutes");
const userRouter = require("./routes/userRoutes");

const app = express();

//middleware
console.log(process.env.NODE_ENV);
if (process.env.NODE_ENV === "development") {
  app.use(morgan("dev"));
}
//aita akhon shob file e available

//middleware
app.use(express.json());
app.use(express.static(`${__dirname}/public`));

app.use((req, res, next) => {
  console.log("Hello from the middlerare");
  next();
});

app.use("/api/v1/tours", tourRouter);
app.use("api/v1/users", userRouter);

module.exports = app;
```

Server.js

```
const dotenv = require("dotenv");
dotenv.config({ path: "./config.env" });

//config er por path
const app = require("./app");

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});
```

Config.js
NODE_ENV=development
PORT=3000
USER = mila
PASSWORD =123456

Postman: get all tour
Terminal output:
development
App running on port 3000...
Hello from the middlerare
GET /api/v1/tours 200 12.191 ms - 8881

NODE_ENV=production nodemon server.js dile kaj korbe na.Nicher ta dite hobe…

```
"scripts": {
   "start:dev": "nodemon server.js",
   "start:prod": "SET NODE_ENV=production && nodemon server.js"
 },
```

Run korer jonno: npm run start:prod & npm run start:dev

//

ESLint:

EsLint basically a program that constantly scan our code and finds potential coding errors.Controll Bad coding practices & it's vary congigurable.

Install EsLint, Prettier from VS code. Then again it from install terminal.

Akhon amder kisu js stype guide dorkar ja amra follow korbo. So most popular style guide = airbnb

Ar jonno akhon eslint config korbo.

NPM INSTALL:

npm i eslint prettier eslint-config-prettier eslint-plugin-prettier eslint-config-airbnb eslint-plugin-node eslint-plugin-import eslint-plugin-jsx-a11y eslint-plugin-react - -save-dev

React code na likle o ai khane react dorkar karon airbnb aitar uppor depent.

Er por onno kono project a use korte config file copy kore niya gele e hobe.

Ai process globaly kaj korbe na.

```
"devDependencies": {
  "eslint": "^8.1.0",
  "eslint-config-airbnb": "^18.2.1",
  "eslint-config-prettier": "^8.3.0",
  "eslint-plugin-import": "^2.25.2",
  "eslint-plugin-jsx-a11y": "^6.4.1",
  "eslint-plugin-node": "^11.1.0",
  "eslint-plugin-prettier": "^4.0.0",
  "eslint-plugin-react": "^7.26.1",
  "prettier": "^2.4.1"
```

```
  }
}
```

Now config file for both prettier & eslint……..

.prettierrc

Website: https://eslint.org/docs/rules/

Age package.json file a copy kore er por npm Install dile shob install hoiya jabe