Page 1: Introduction to Express.js

1. What is Express.js?

- A **minimal and flexible Node.js framework** for building web applications and APIs.
- Handles routing, middleware, and HTTP requests/responses easily.

2. Installation

```
npm init -y

npm install express
```

3. Basic Server Setup

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

const app = express();

const PORT = 3000;


app.get('/', (req, res) => {

  res.send('Welcome to Express!');

});


app.listen(PORT, () => {

  console.log(`Server running on port ${PORT}`);

});
```

4. Key Features

- Middleware support
- Routing
- Template engines
- Easy integration with databases

⚙️ **Page 2: Routing and Middleware**

1. Routing

```
app.get('/users', (req, res) => {
```

```
  res.send('List of users');
});

app.post('/users', (req, res) => {
  res.send('User created');
});
```

2. Route Parameters

```
app.get('/users/:id', (req, res) => {
  res.send(`User ID: ${req.params.id}`);
});
```

3. Middleware

- Middleware are functions that execute during the request–response cycle.

```
app.use(express.json()); // built-in middleware
```

Custom Middleware

```
app.use((req, res, next) => {
  console.log('Middleware executed');
  next();
});
```

4. Error Handling

```
app.use((err, req, res, next) => {
  res.status(500).send('Something broke!');
});
```

💾 Page 3: Building an API

1. CRUD Operations Example

```
let books = [];

app.post('/books', (req, res) => {
```

```
  books.push(req.body);
  res.send('Book added');
});

app.get('/books', (req, res) => {
  res.json(books);
});

app.put('/books/:id', (req, res) => {
  const { id } = req.params;
  books[id] = req.body;
  res.send('Book updated');
});

app.delete('/books/:id', (req, res) => {
  const { id } = req.params;
  books.splice(id, 1);
  res.send('Book deleted');
});
```

2. Connecting with MongoDB

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/mydb')
  .then(() => console.log('Connected to MongoDB'));
```

3. Best Practices

- Use dotenv for environment variables
- Structure files (routes, controllers, models)
- Use try-catch blocks for async operations
- Handle 404 and errors gracefully