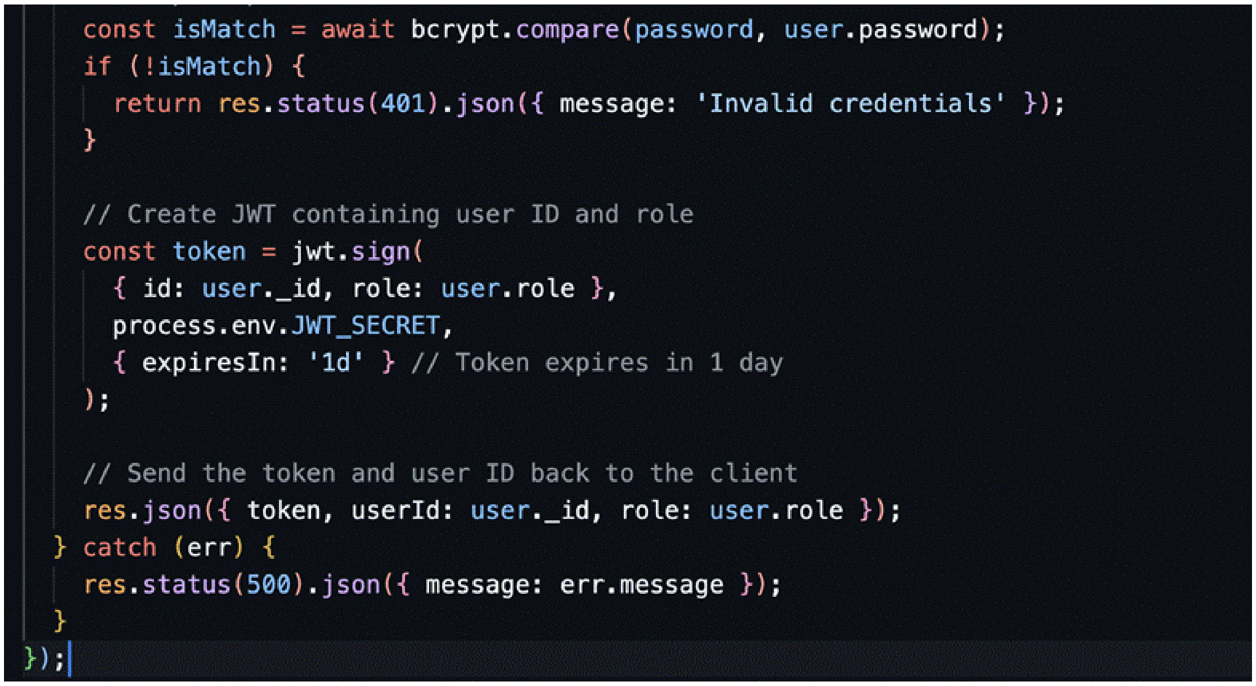
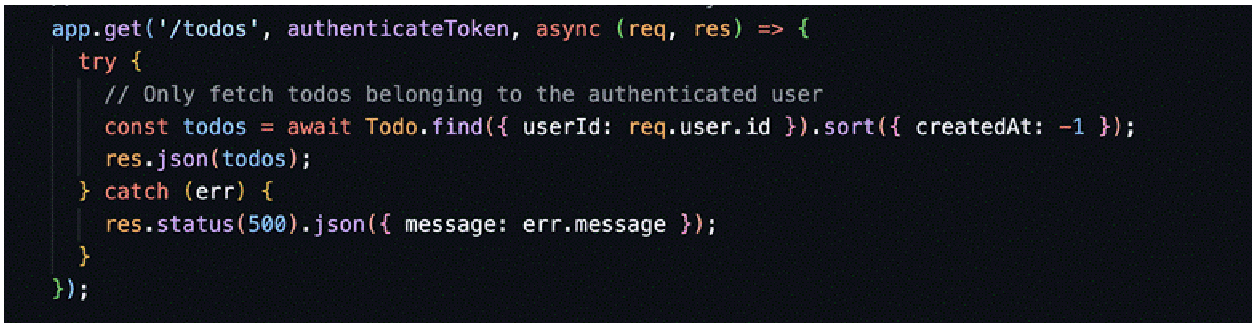
Experiment 6

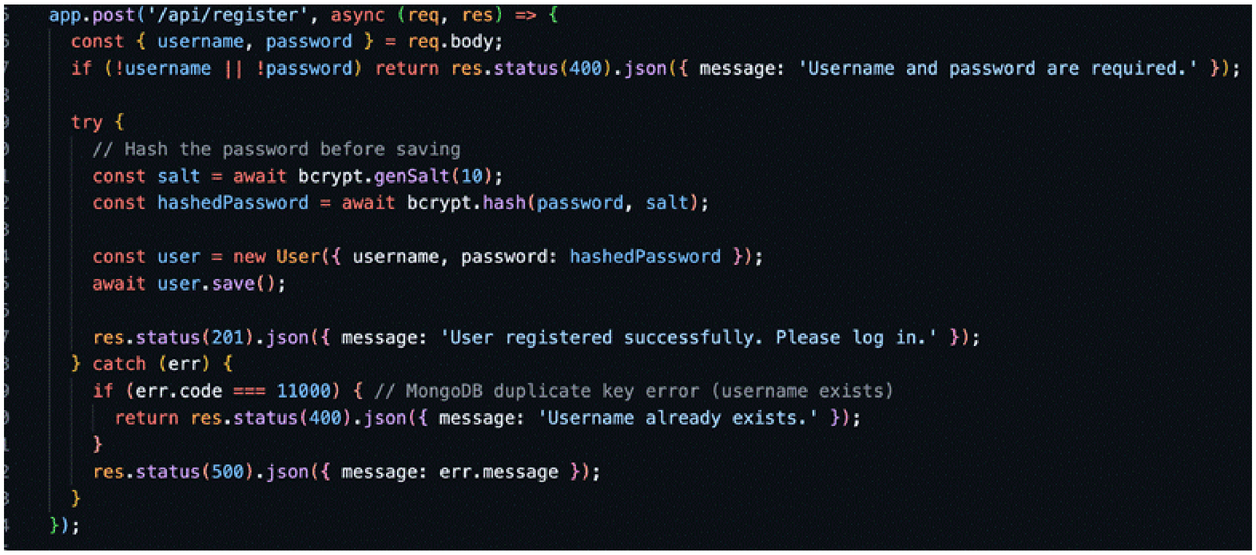
Code :



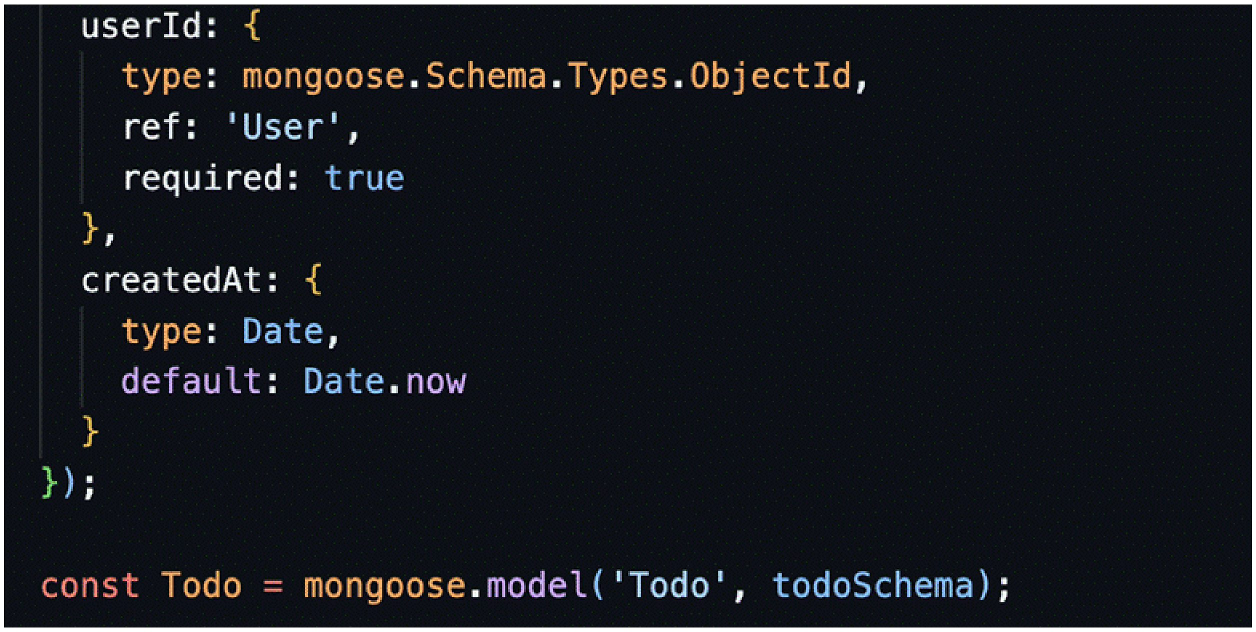
```
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }

    // Create JWT containing user ID and role
    const token = jwt.sign(
      { id: user._id, role: user.role },
      process.env.JWT_SECRET,
      { expiresIn: '1d' } // Token expires in 1 day
    );

    // Send the token and user ID back to the client
    res.json({ token, userId: user._id, role: user.role });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```



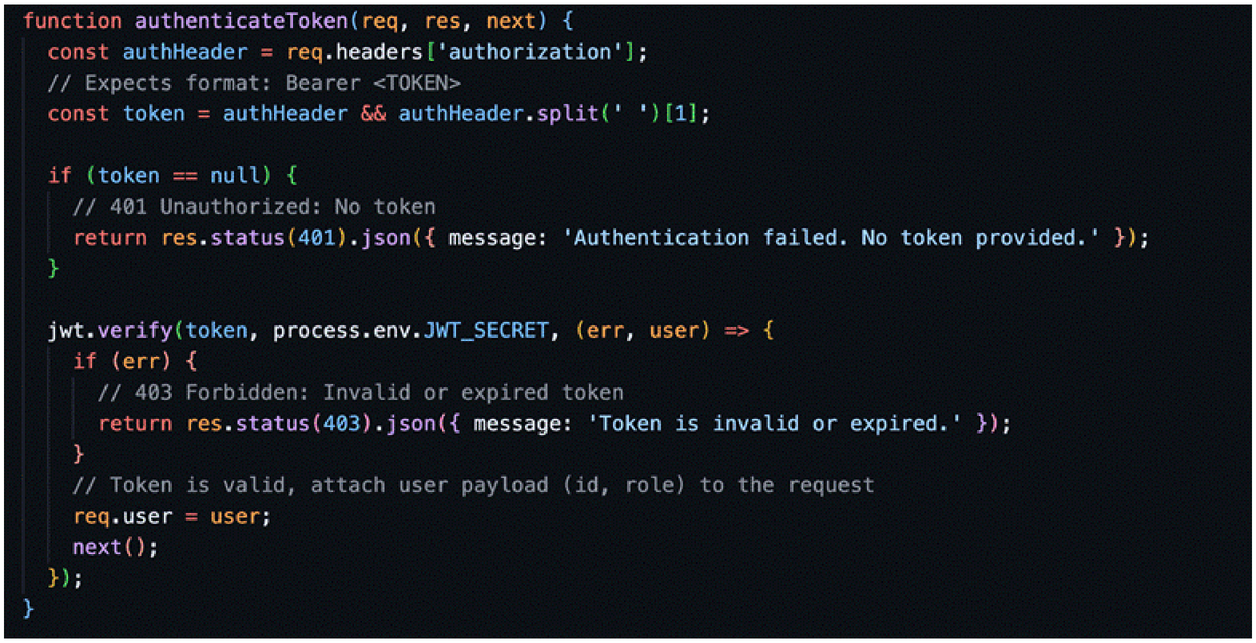
```
app.get('/todos', authenticateToken, async (req, res) => {
  try {
    // Only fetch todos belonging to the authenticated user
    const todos = await Todo.find({ userId: req.user.id }).sort({ createdAt: -1 });
    res.json(todos);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});
```



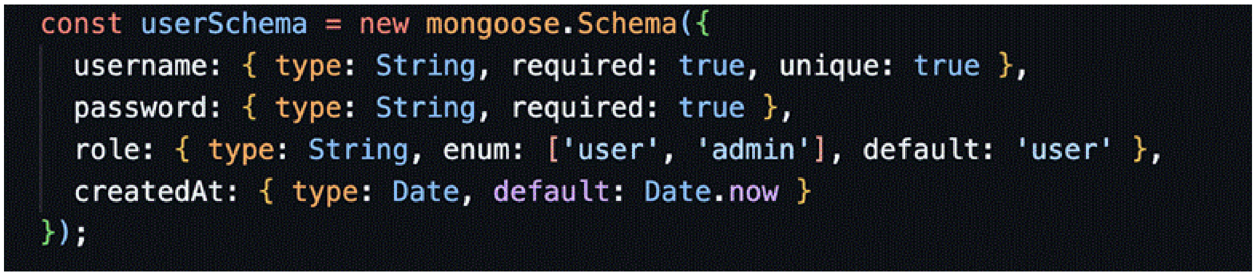
```
app.post('/api/register', async (req, res) => {
  const { username, password } = req.body;
  if (!username || !password) return res.status(400).json({ message: 'Username and password are required.' });

  try {
    // Hash the password before saving
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(password, salt);

    const user = new User({ username, password: hashedPassword });
    await user.save();

    res.status(201).json({ message: 'User registered successfully. Please log in.' });
  } catch (err) {
    if (err.code === 11000) { // MongoDB duplicate key error (username exists)
      return res.status(400).json({ message: 'Username already exists.' });
    }
    res.status(500).json({ message: err.message });
  }
});
```



```
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

const Todo = mongoose.model('Todo', todoSchema);
```



```
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  // Expects format: Bearer <TOKEN>
  const token = authHeader && authHeader.split(' ')[1];

  if (token == null) {
    // 401 Unauthorized: No token
    return res.status(401).json({ message: 'Authentication failed. No token provided.' });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) {
      // 403 Forbidden: Invalid or expired token
      return res.status(403).json({ message: 'Token is invalid or expired.' });
    }
    // Token is valid, attach user payload (id, role) to the request
    req.user = user;
    next();
  });
}
```



```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['user', 'admin'], default: 'user' },
  createdAt: { type: Date, default: Date.now }
});
```

Output :

Fig : login page

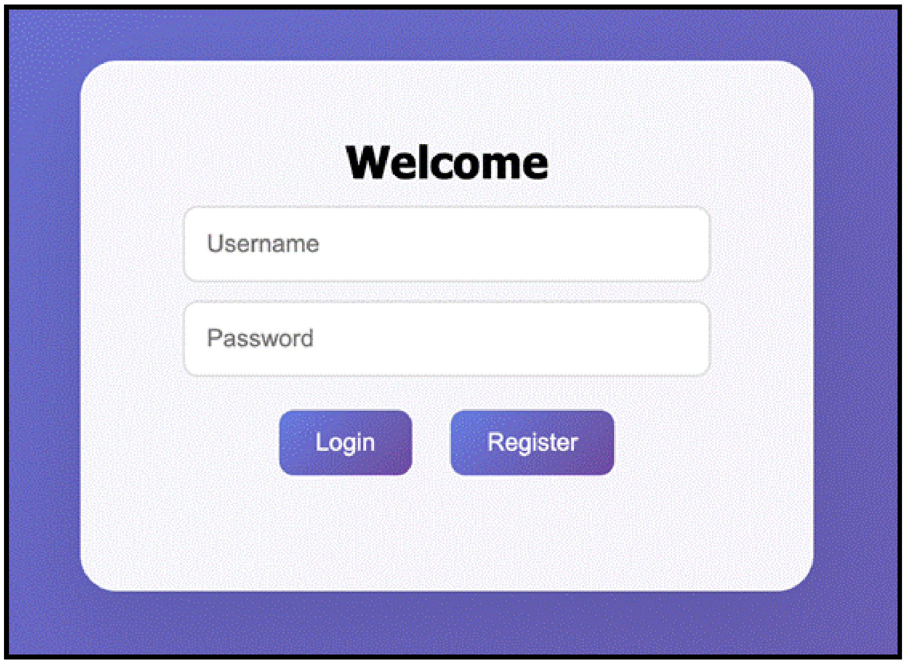
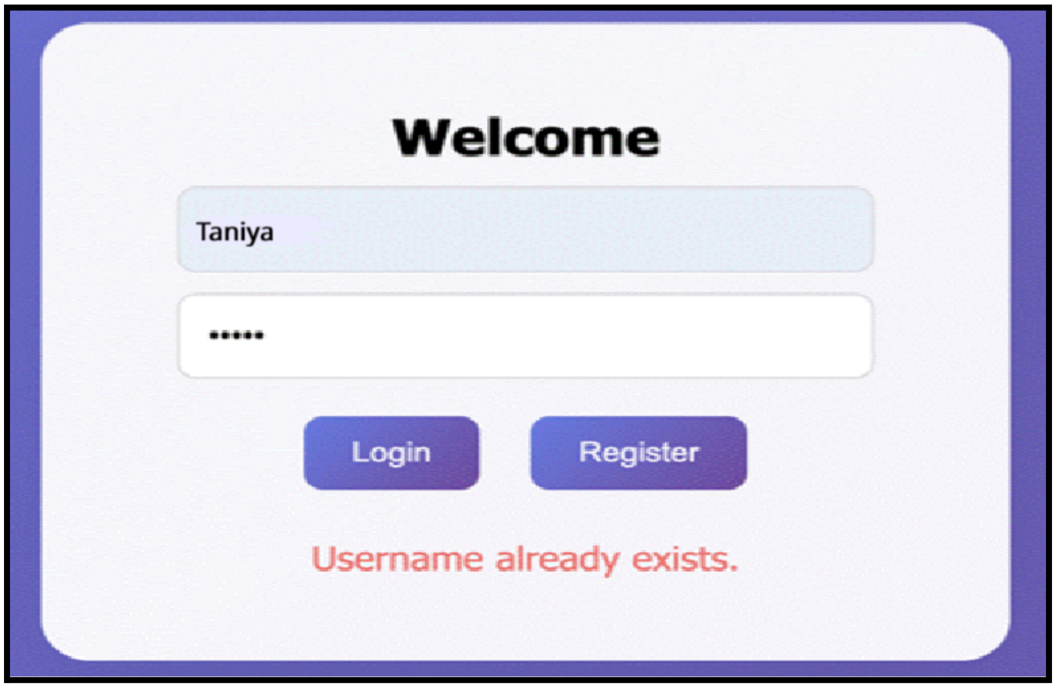
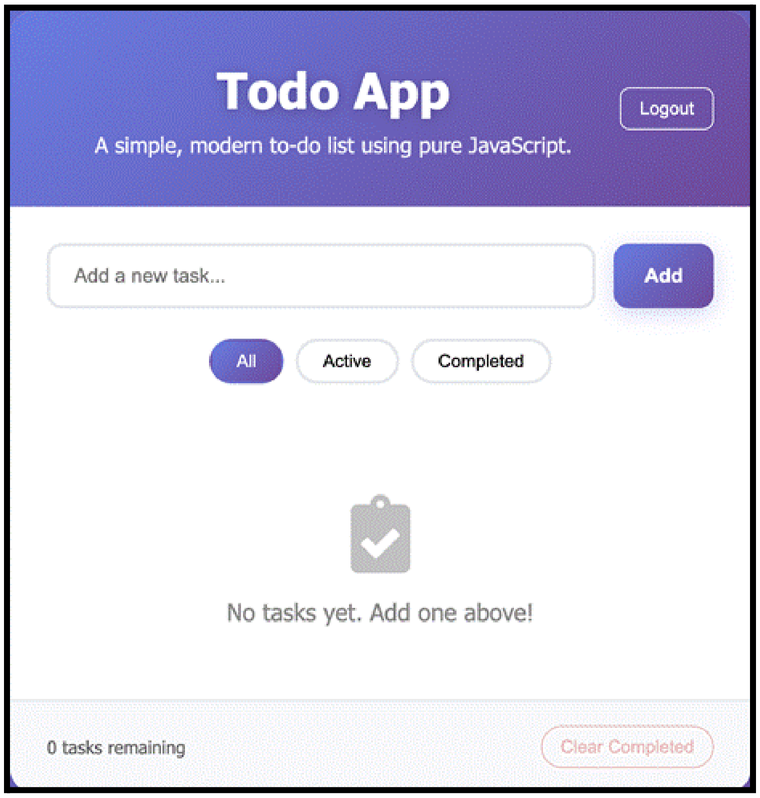


Fig: register



Username already exists.

Fig: Enhanced UI


Todo App
A simple, modern to-do list using pure JavaScript.
Logout
Add a new task...
Add
All
Active
Completed
No tasks yet. Add one above!
0 tasks remaining
Clear Completed