

ISSUES TRACKER

TEAM ID: NM2023TMID01436

Project Link: [naanmudhalvan-SI/IBM--14003-1682490056: The IssueTracker: A Reliable Complaint Management System for Improved Customer Service \(github.com\)](https://github.com/naanmudhalvan-SI/IBM--14003-1682490056-The-IssueTracker-A-Reliable-Complaint-Management-System-for-Improved-Customer-Service)

1.INTRODUCTION

An issue tracker, also known as a bug tracker or ticketing system, is a software tool used by individuals and teams to manage and track issues, bugs, and tasks throughout a project's lifecycle. It provides a centralized platform to log, organize, and prioritize various types of work items, allowing for efficient collaboration and streamlined workflow.

1.1Purpose

Users can create new issues or tickets to report bugs, problems, or new features. Each issue typically includes relevant details such as a title, description, priority, status, assignee, and any associated attachments or comments. The tracker allows teams to keep a record of all issues in a structured and searchable manner. Each issue is assigned a unique identifier, making it easy to reference and track its progress over time.

2. LITERATURE SURVEY

In the context of an issue tracker, a literature survey refers to the process of researching and documenting relevant existing literature, studies, articles, and other sources of information related to a specific issue or topic. It involves conducting a comprehensive review of the existing body of knowledge to gain a deeper understanding of the subject matter. In an issue tracker, a literature survey may be performed as part of problem analysis or issue resolution. It helps the individuals or teams involved in addressing the issue to gather insights, learn from previous experiences, and understand the existing solutions or approaches that have been proposed or implemented by others.

2.1 Existing problem

Issue trackers can be complex tools with extensive features, making them difficult for new users to navigate. If the interface is not intuitive or lacks proper documentation, it can hinder user adoption and productivity. Some issue trackers may not support flexible workflows or customization options, forcing users to adapt their processes to fit the system rather than vice versa. This can lead to inefficiencies and frustration when trying to manage and track issues effectively.

2.2 References

Trello is a visual collaboration tool that allows users to create boards, lists, and cards to manage projects and track issues. It offers a flexible interface and integrates with other tools for enhanced functionality.

Website: <https://trello.com>

GitHub Issues is an issue tracking tool integrated with the GitHub code repository platform. It allows developers to create, track, and manage issues directly within their code repositories. Website:

<https://github.com/features/issues>

2.3 Problem Statement Definition

The problem statement should provide a clear and concise description of the issue or problem that is being reported. It should explain the symptoms or behavior observed, the impact it has on users or the system, and any relevant context or background information. If possible, the problem statement should indicate the steps or conditions required to reproduce the issue. This information helps developers and testers to understand and replicate the problem, facilitating the debugging and resolution process.

3. IDEATION & PROPOSED SOLUTION

Users may propose different approaches, alternative features, or changes in functionality that could potentially solve the problem or enhance the system's capabilities. These ideas are often captured as comments, descriptions, or attachments within the issue tracker, allowing stakeholders to document and share their suggestions.

3.1 Empathy Map Canvas

3.3 Proposed Solution

A proposed solution refers to a suggested resolution or approach to address a specific problem or task mentioned in an issue or ticket. When users encounter a bug, request a new feature, or report an issue, they often include detailed information about the problem they are experiencing or the desired outcome they would like to achieve. The proposed solution is a response to that issue and outlines the steps, changes, or fixes that could potentially resolve the problem or fulfill the request. It serves as a recommendation or suggestion to the development team or individuals responsible for addressing the issue.

4. REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design

4.1 Functional requirement

- prediction
- percentage of chance
- eligibility

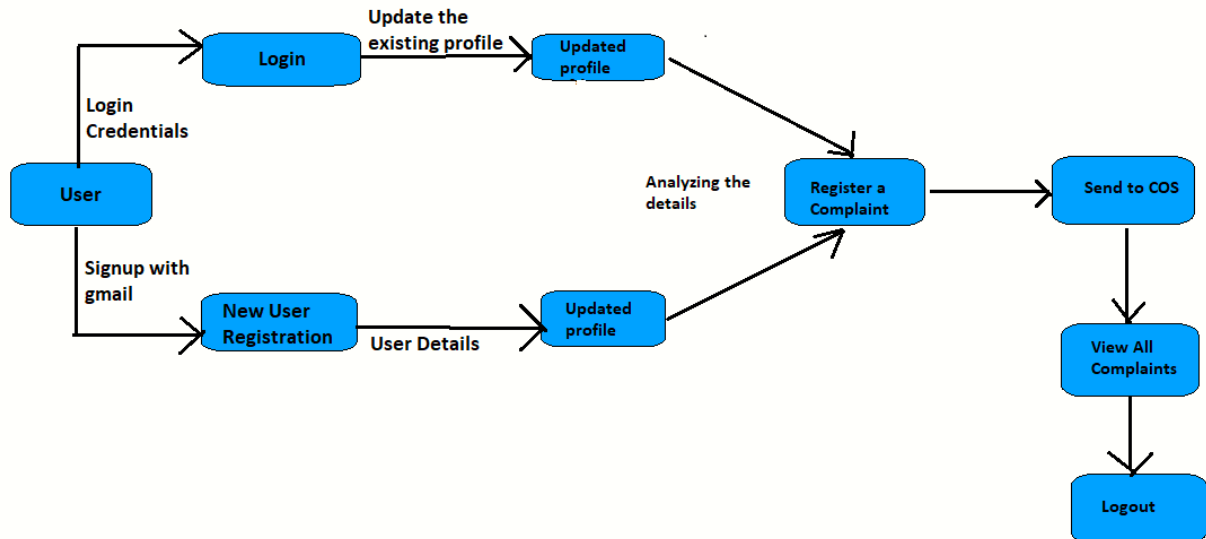
4.2 Non-Functional requirements

- Speed
- Security
- Portability
- Compatibility
- Capacity
- Reliability
- Environment

5. PROJECT DESIGN

5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

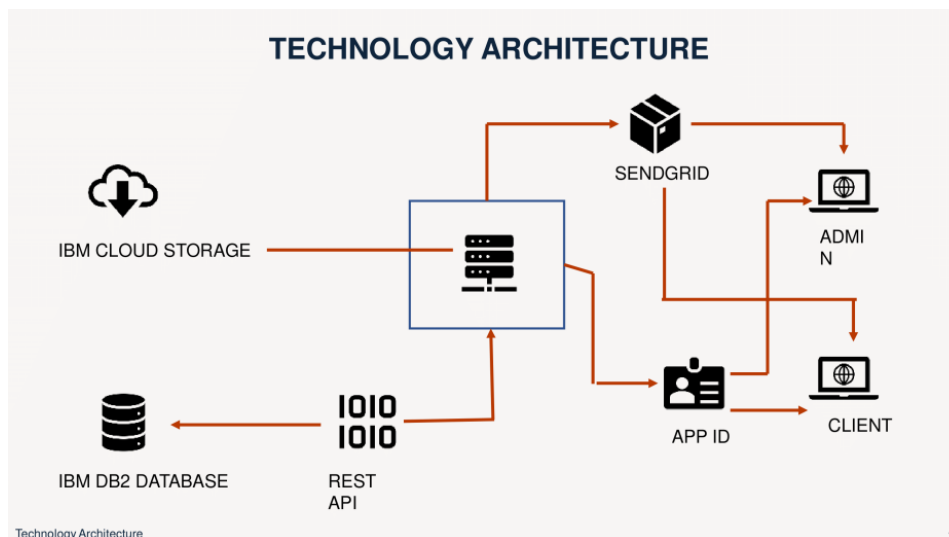


5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.

Provide specifications according to which the solution is defined, managed, and delivered.



5.3 User Stories

A user story is an informal, natural language description of features of a software system. They are written from the perspective of an end user or user of a system, and may be recorded on index cards, Post-it notes, or digitally in project management software.[1] Depending on the project, user stories may be written by different stakeholders like client, user, manager, or development team.

6. PROJECT PLANNING & SCHEDULING

TECHNOLOGY ARCHITECTURE			
S.NO	COMPONENT	DESCRIPTION	TECHNOLOGY
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL etc
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Technology Architecture

5

APPLICATION CHARACTERISTICS			
S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	python flask
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g., encryption, intrusion detection software, antivirus, firewalls
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	supports higher workloads without any fundamental changes to it.
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	High availability enables your IT infrastructure to continue functioning even when some of its components fail.
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Performance technology, therefore, is a field of practice that uses various tools, processes, and ideas in a scientific, systematic manner to improve the desired outcomes of individuals and organizations.

Technology Architecture

6

7.PROJECT CODE

7.1 APP CODE

```
File Edit Selection View Go Run Terminal Help
main.py - Visual Studio Code

main.py 5 X
D:\> IBM > started > main.py > signup
1 from flask import Flask, render_template, request, redirect, url_for
2 import ibm_db
3 import ibm_boto3
4 from ibm_botocore.client import Config, ClientError
5 import os
6
7 app = Flask(__name__, template_folder='templateFiles', static_folder='staticFiles')
8 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgu0lqde00.databases.appdomain.cloud;PORT=30699;SEC
9 useremail='
10 cos_endpoint="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
11 cos_apikey="Yis1PT0Vne3j0DR2j5ePHEfLmrtApQsav1St9IoSELMn"
12 cos_crn="crn:v1:bluemix:public:cloud-object-storage:global:a/4ba25fea32e046cdb8e6cbe2ad24f6ca:d144147f-700e-491b-b7d6-c07c8ba51fa2::"
13 cos=ibm_boto3.client('s3',ibm_api_key_id=cos_apikey,ibm_service_instance_id=cos_crn,endpoint_url=cos_endpoint,config= Config(signature_version="o
14
15 @app.route('/reg',methods=['GET','POST'])
16 @app.route('/',methods=['GET','POST'])
17 def index():
18     return render_template('reg.html')
19
20
21 @app.route('/sign',methods=['GET','POST'])
22 def signup():
23     if request.method=="POST":
24         check=True
25         name=request.form['txt']
26         name=name.lower()
27         email=request.form['email']
28         email=email.lower()
29         password=request.form['pswd']
30         sql = "SELECT * FROM TEST"
31         stmt = ibm_db.exec_immediate(conn, sql)
32         dictionary = ibm_db.fetch_both(stmt)
33         msgb=""
```

```
File Edit Selection View Go Run Terminal Help
main.py - Visual Studio Code

main.py 5 X
D:\> IBM > started > main.py > signup
31 stmt = ibm_db.exec_immediate(conn, sql)
32 dictionary = ibm_db.fetch_both(stmt)
33 msgb=""
34 while dictionary != False:
35     if email==dictionary[1]:
36         check=False
37         msgb="Already Registered"
38         msgb="Click to Login"
39         break
40     dictionary = ibm_db.fetch_both(stmt)
41 if check==True:
42     sql = "INSERT INTO TEST VALUES (?,?,?)"
43     stmt = ibm_db.prepare(conn,sql)
44     ibm_db.bind_param(stmt,1,str(username))
45     ibm_db.bind_param(stmt,2,str(email))
46     ibm_db.bind_param(stmt,3,str(password))
47     ibm_db.execute(stmt)
48     msgb="Click here to Login"
49     msgb="Successfully Registered"
50     return render_template('regerror.html',msg=msg,msgb=msgb)
51 else:
52     return render_template('regerror.html',msg=msg,msgb="Click to Login")
53
54
55
56 @app.route('/log',methods=['GET','POST'])
57 def login():
58     global useremail
59     if request.method=="POST":
60         check=True
61         email=request.form['email']
62         useremail=email.lower()
63         password=request.form['pswd']
```

The image displays two screenshots of a Visual Studio Code editor window, showing Python code for a web application. The editor is titled "main.py - Visual Studio Code".

Top Screenshot: The code defines a login function and a home route. The login function checks if the user email is "admin@123.com" and the password is "admin". If correct, it renders "homeadmin.html". Otherwise, it checks a database for the user. If found, it renders "homeadmin.html". If not found, it renders "regerror.html" with a message. The home route renders "reg.html" if the user is not logged in, otherwise "homepage.html".

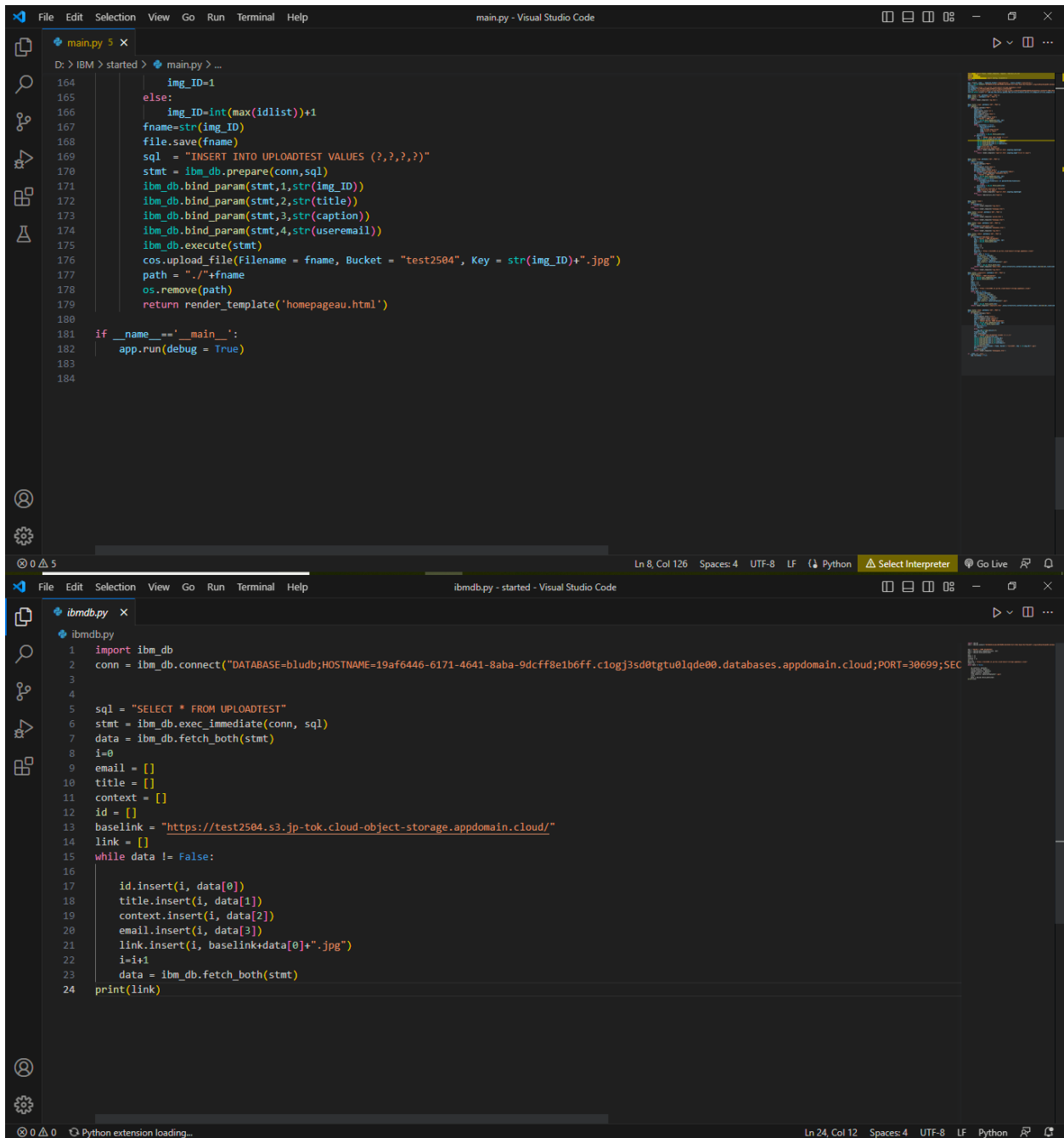
```
61 email=request.form['email']
62 useremail=email.lower()
63 password=request.form['pswd']
64 if useremail=="admin@123.com" and password=="admin":
65     return render_template("homeadmin.html")
66 sql = "SELECT * FROM TEST"
67 stmt = ibm_db.exec_immediate(conn, sql)
68 dictionary = ibm_db.fetch_both(stmt)
69 while dictionary != False:
70     if useremail==dictionary[1] and password==dictionary[2]:
71         check=False
72         break
73     dictionary = ibm_db.fetch_both(stmt)
74 if check==True:
75     msg="Incorrect Username or Password"
76     msgb="Click to Try again"
77     return render_template("regerror.html",msg=msg,msgb=msgb)
78 else:
79     return redirect(url_for("home"))
80
81
82
83 @app.route('/home')
84 def home():
85     if useremail=='':
86         return render_template("reg.html")
87     else:
88         return render_template("homepage.html")
89
90 @app.route('/upload',methods=['GET','POST'])
91 def upload():
92     if useremail=='':
```

Bottom Screenshot: The code defines an upload function, a page route, and an admin route. The upload function renders "upload.html" if the user is not logged in, otherwise "homepage.html". The page route renders "homeadmin.html" if the user is "admin@123.com", otherwise "reg.html". The admin route checks if the user is "admin@123.com" and if the password is correct, it inserts data into a database. If not correct, it renders "reg.html".

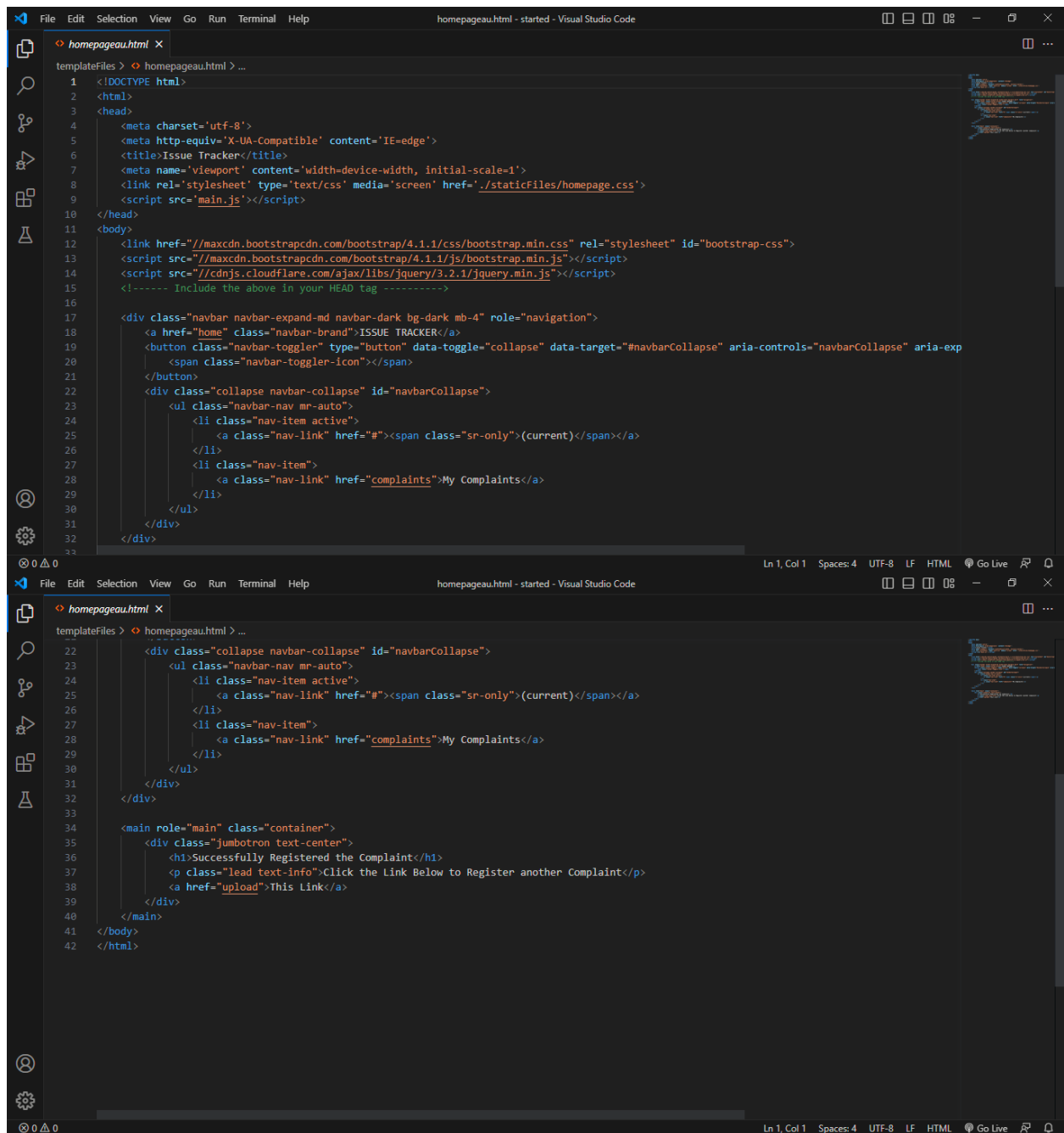
```
93     return render_template("upload.html")
94 else:
95     return render_template("homepage.html")
96
97 @app.route('/page',methods=['GET','POST'])
98 def page():
99     if useremail=="admin@123.com":
100         return render_template("homeadmin.html")
101     else:
102         return render_template("reg.html")
103
104 @app.route('/admin',methods=['GET','POST'])
105 def admin():
106     if useremail=="admin@123.com":
107         sql = "SELECT * FROM UPLOADTEST"
108         stmt = ibm_db.exec_immediate(conn, sql)
109         data = ibm_db.fetch_both(stmt)
110         i=0
111         email = []
112         title = []
113         context = []
114         id = []
115         baselink = "https://test2504.s3.jp-tok.cloud-object-storage.appdomain.cloud/"
116         link = []
117         while data != False:
118             id.insert(i, data[0])
119             title.insert(i, data[1])
120             context.insert(i, data[2])
121             email.insert(i, data[3])
```



```
File Edit Selection View Go Run Terminal Help main.py - Visual Studio Code
main.py 5 X
D: > IBM > started > main.py > ...
118 id.insert(i, data[0])
119 title.insert(i, data[1])
120 context.insert(i, data[2])
121 email.insert(i, data[3])
122 link.insert(i, baselink+data[0]+".jpg")
123 i=i+1
124 data = ibm_db.fetch_both(stmt)
125 return render_template("admin.html",id=id,title=title,context=context,email=email,len=len(id),link=link)
126 else:
127     return render_template("reg.html")
128
129 @app.route('/complaints',methods=['GET','POST'])
130 def complaints():
131     sql = "SELECT * FROM UPLOADTEST"
132     stmt = ibm_db.exec_immediate(conn, sql)
133     data = ibm_db.fetch_both(stmt)
134     i=0
135     email = []
136     title = []
137     context = []
138     id = []
139     baselink = "https://test2504.s3.jp-tok.cloud-object-storage.appdomain.cloud/"
140     link = []
141     while data != False:
142         if useremail==data[3]:
143             id.insert(i, data[0])
144             title.insert(i, data[1])
145             context.insert(i, data[2])
146             email.insert(i, data[3])
147             link.insert(i, baselink+data[0]+".jpg")
148             i=i+1
149         data = ibm_db.fetch_both(stmt)
150     return render_template("complaints.html",id=id,title=title,context=context,email=email,len=len(id),link=link)
151
152
153 @app.route('/home',methods=['GET','POST'])
154 def uploadfile():
155     if request.method=="POST":
156         check=True
157         title=request.form['title']
158         caption=request.form['caption']
159         file = request.files['images']
160         sql = "SELECT MAX(ID) FROM UPLOADTEST"
161         stmt = ibm_db.exec_immediate(conn, sql)
162         idlist = ibm_db.fetch_tuple(stmt)
163         if idlist==False:
164             img_ID=1
165         else:
166             img_ID=int(max(idlist))+1
167         fname=str(img_ID)
168         file.save(fname)
169         sql = "INSERT INTO UPLOADTEST VALUES (?,?,,?)"
170         stmt = ibm_db.prepare(conn,sql)
171         ibm_db.bind_param(stmt,1,str(img_ID))
172         ibm_db.bind_param(stmt,2,str(title))
173         ibm_db.bind_param(stmt,3,str(caption))
174         ibm_db.bind_param(stmt,4,str(useremail))
175         ibm_db.execute(stmt)
176         cos.upload_file(Filename = fname, Bucket = "test2504", Key = str(img_ID)+".jpg")
177         path = "./"+fname
178         os.remove(path)
179         return render_template("homepageau.html")
180
```

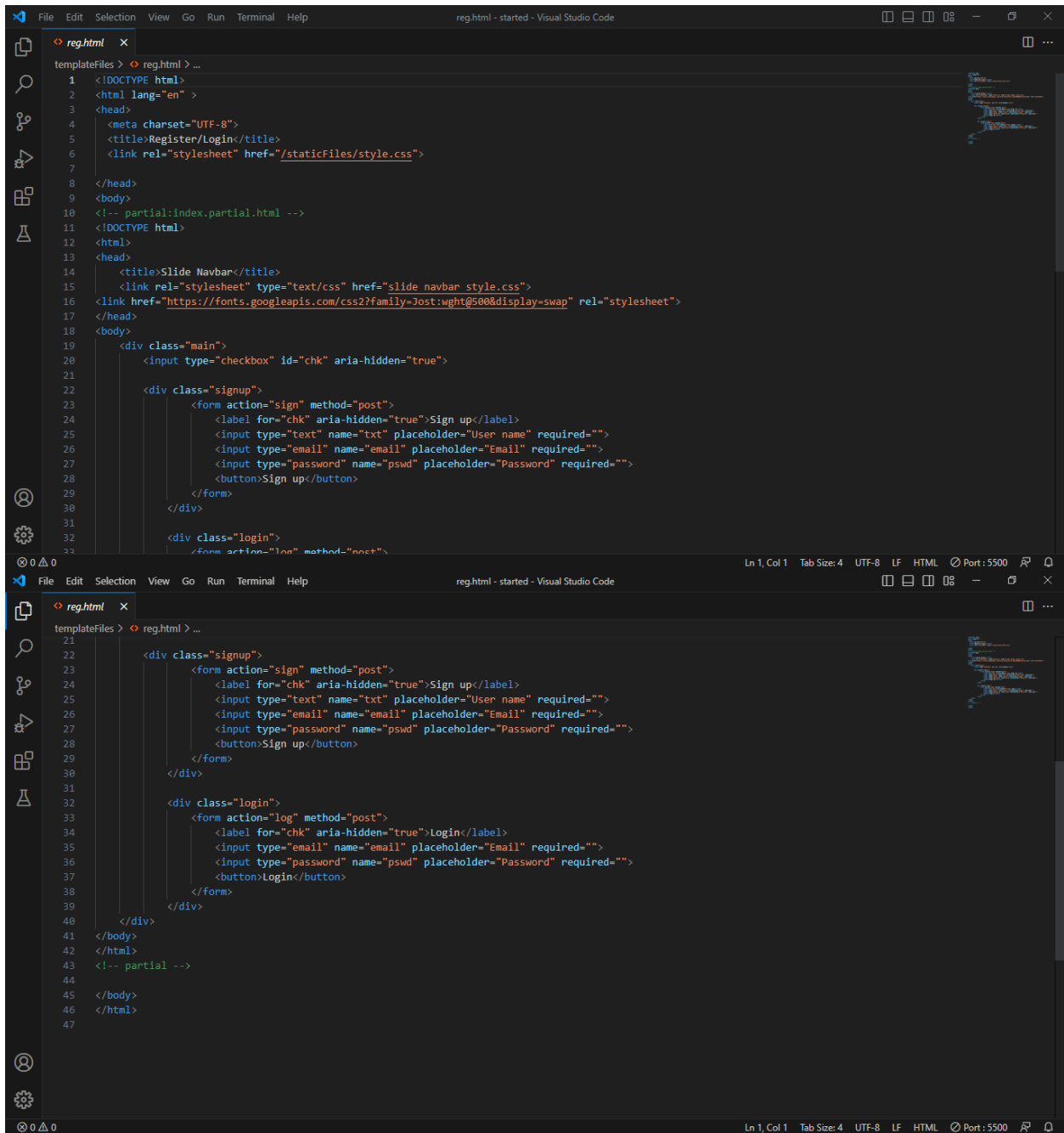


7.2 INDEX



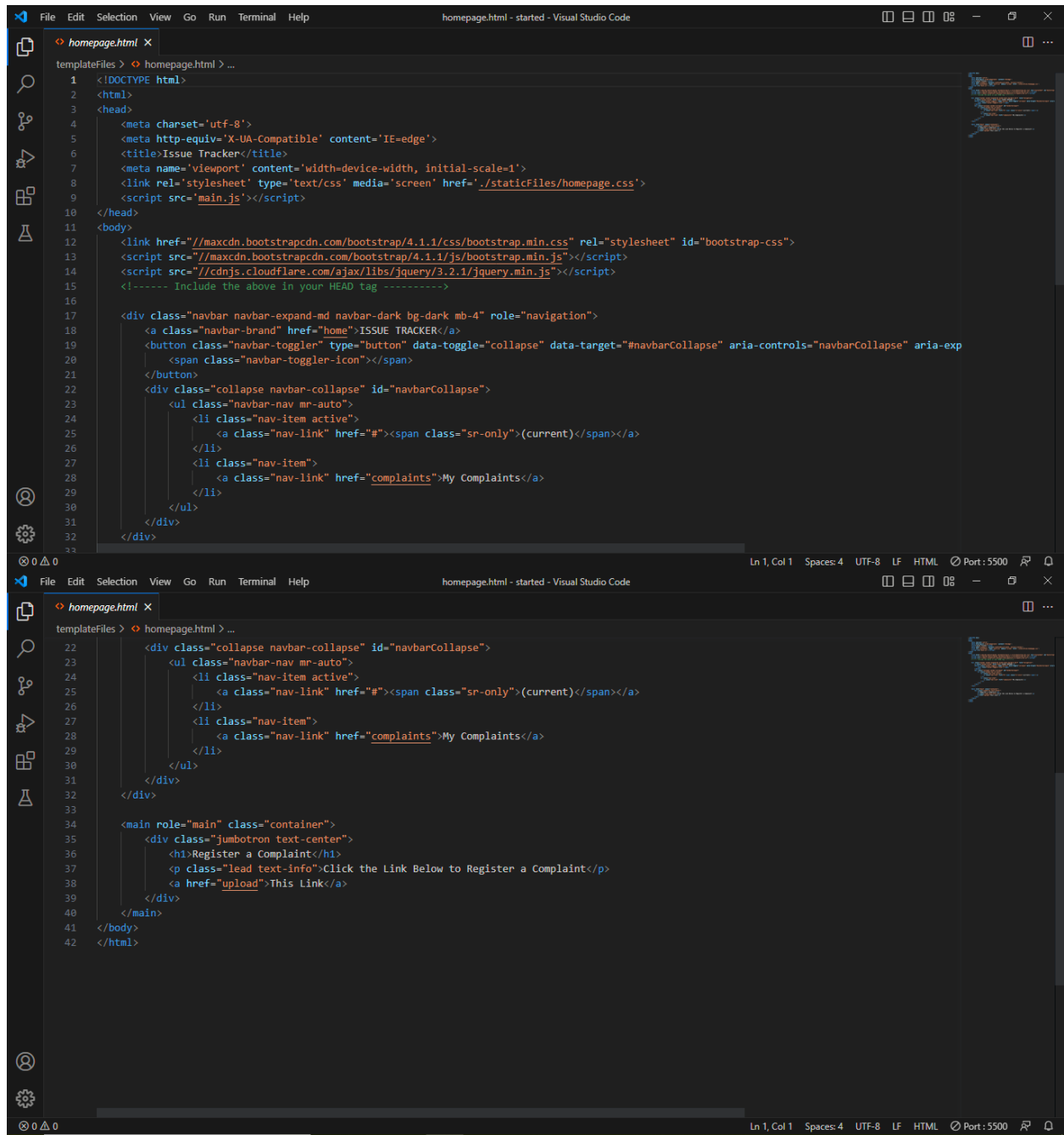
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6   <title>Issue Tracker</title>
7   <meta name='viewport' content='width=device-width, initial-scale=1'>
8   <link rel='stylesheet' type='text/css' media='screen' href='../staticFiles/homepage.css'>
9   <script src='main.js'></script>
10 </head>
11 <body>
12   <link href='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css' rel='stylesheet' id='bootstrap-css'>
13   <script src='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js'></script>
14   <script src='//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>
15   <!-- Include the above in your HEAD tag -->
16
17   <div class='navbar navbar-expand-md navbar-dark bg-dark mb-4' role='navigation'>
18     <a href='home' class='navbar-brand'>ISSUE TRACKER</a>
19     <button class='navbar-toggler' type='button' data-toggle='collapse' data-target='#navbarCollapse' aria-controls='navbarCollapse' aria-exp
20       <span class='navbar-toggler-icon'></span>
21     </button>
22     <div class='collapse navbar-collapse' id='navbarCollapse'>
23       <ul class='navbar-nav mr-auto'>
24         <li class='nav-item active'>
25           <a class='nav-link' href='#'><span class='sr-only'>(current)</span></a>
26         </li>
27         <li class='nav-item'>
28           <a class='nav-link' href='complaints'>My Complaints</a>
29         </li>
30       </ul>
31     </div>
32   </div>
33
34   <main role='main' class='container'>
35     <div class='jumbotron text-center'>
36       <h1>Successfully Registered the Complaint</h1>
37       <p class='lead text-info'>Click the Link Below to Register another Complaint</p>
38       <a href='upload'>This Link</a>
39     </div>
40   </main>
41 </body>
42 </html>
```

7.3 REGISTRATION



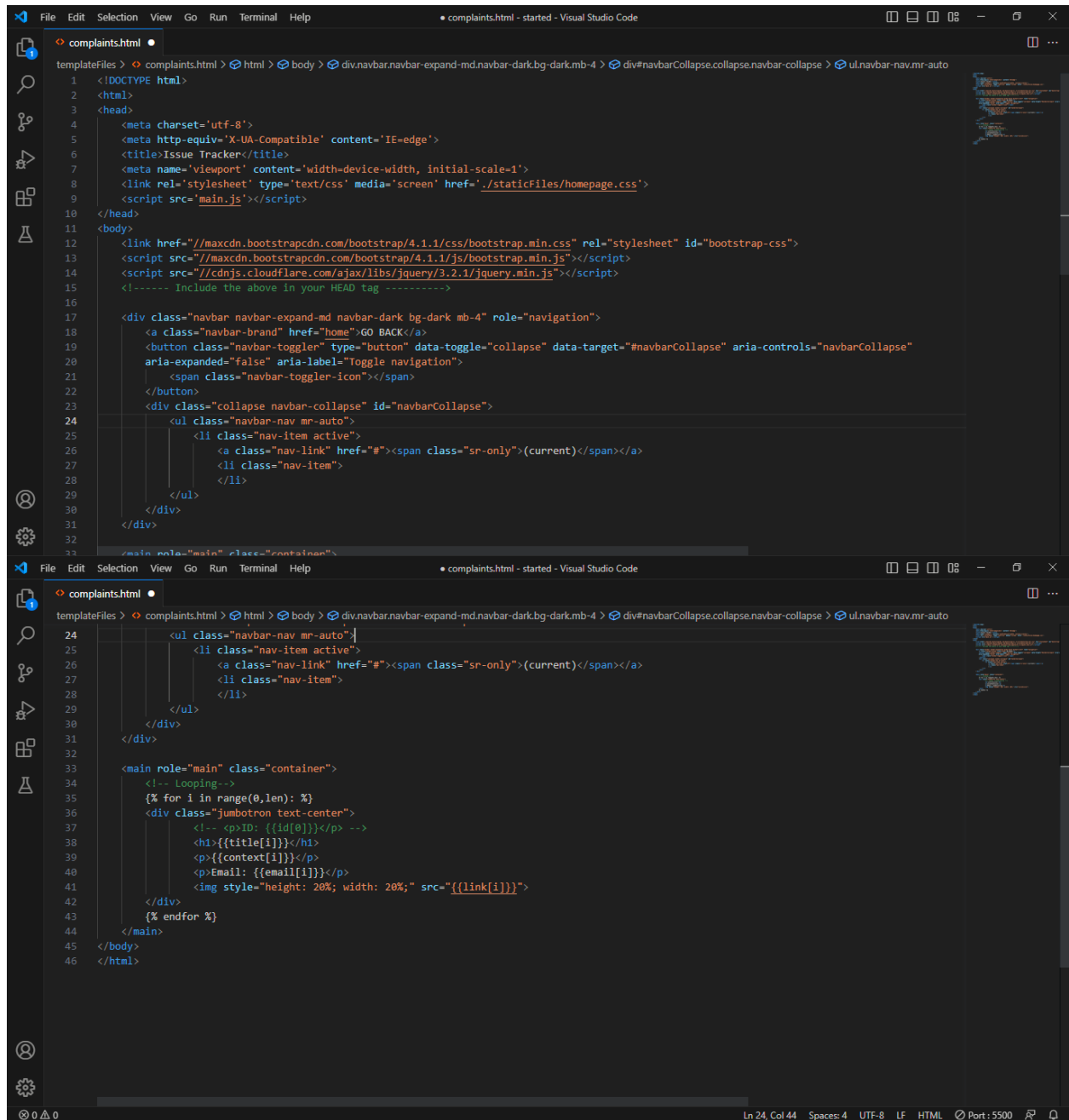
```
reg.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Register/Login</title>
6   <link rel="stylesheet" href="/staticFiles/style.css">
7 </head>
8 <body>
9 <!-- partial:index.partial.html -->
10 <!DOCTYPE html>
11 <html>
12 <head>
13   <title>Slide Navbar</title>
14   <link rel="stylesheet" type="text/css" href="slide navbar style.css">
15   <link href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap" rel="stylesheet">
16 </head>
17 <body>
18   <div class="main">
19     <input type="checkbox" id="chk" aria-hidden="true">
20
21     <div class="signup">
22       <form action="sign" method="post">
23         <label for="chk" aria-hidden="true">Sign up</label>
24         <input type="text" name="txt" placeholder="User name" required="">
25         <input type="email" name="email" placeholder="Email" required="">
26         <input type="password" name="pswd" placeholder="Password" required="">
27         <button>Sign up</button>
28       </form>
29     </div>
30
31     <div class="login">
32       <form action="log" method="post">
33         <label for="chk" aria-hidden="true">Login</label>
34         <input type="email" name="email" placeholder="Email" required="">
35         <input type="password" name="pswd" placeholder="Password" required="">
36         <button>Login</button>
37       </form>
38     </div>
39   </div>
40 </body>
41 </html>
42 <!-- partial -->
43 </body>
44 </html>
```

7.4 HOMEPAGE



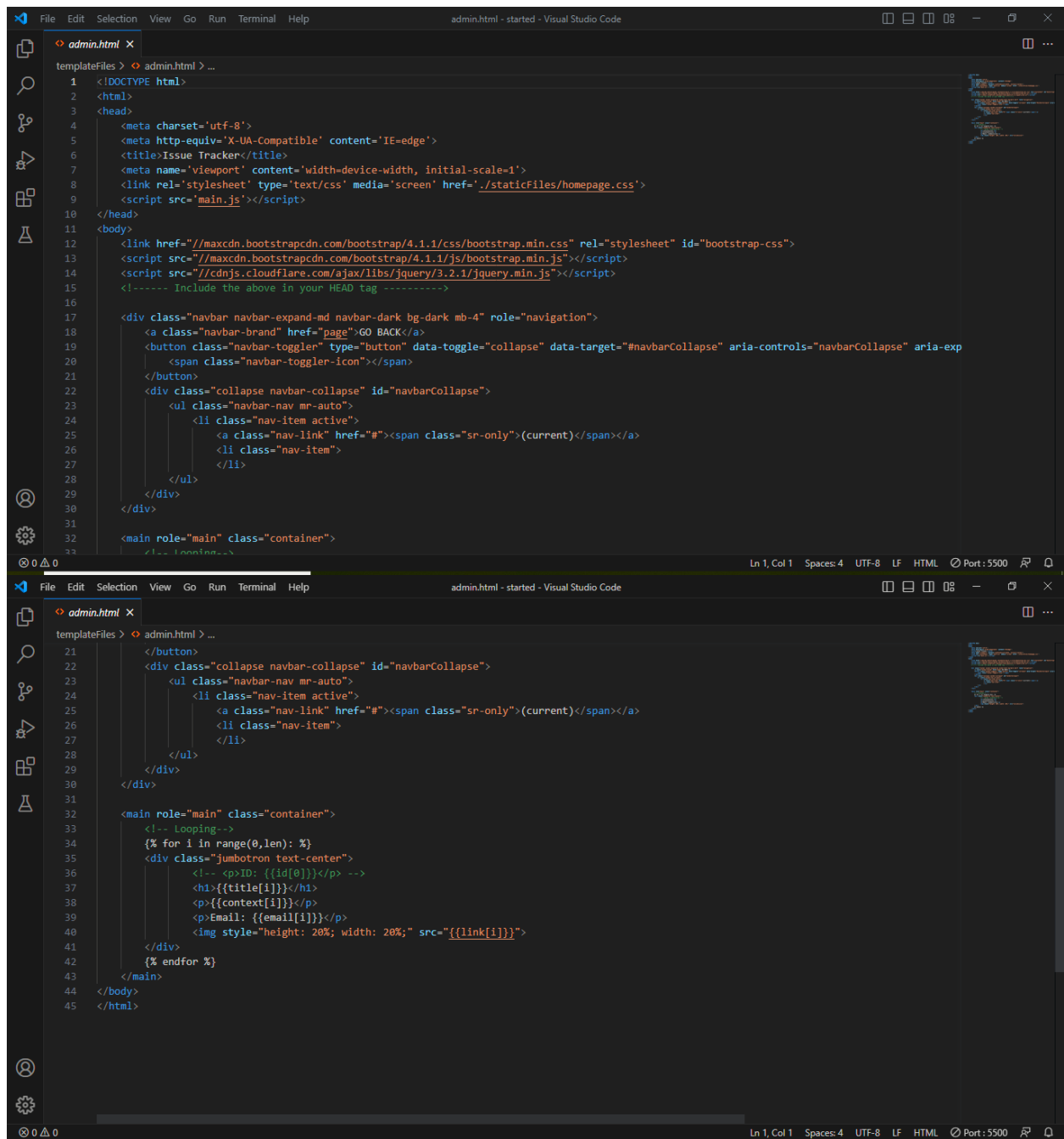
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6   <title>Issue Tracker</title>
7   <meta name='viewport' content='width=device-width, initial-scale=1'>
8   <link rel='stylesheet' type='text/css' media='screen' href='../staticFiles/homepage.css'>
9   <script src='main.js'></script>
10 </head>
11 <body>
12   <link href='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css' rel='stylesheet' id='bootstrap-css'>
13   <script src='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js'></script>
14   <script src='//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>
15   <!-- Include the above in your HEAD tag -->
16
17   <div class='navbar navbar-expand-md navbar-dark bg-dark mb-4' role='navigation'>
18     <a class='navbar-brand' href='home'>ISSUE TRACKER</a>
19     <button class='navbar-toggler' type='button' data-toggle='collapse' data-target='#navbarCollapse' aria-controls='navbarCollapse' aria-exp
20       <span class='navbar-toggler-icon'></span>
21     </button>
22     <div class='collapse navbar-collapse' id='navbarCollapse'>
23       <ul class='navbar-nav mr-auto'>
24         <li class='nav-item active'>
25           <a class='nav-link' href='#'><span class='sr-only'>(current)</span></a>
26         </li>
27         <li class='nav-item'>
28           <a class='nav-link' href='complaints'>My Complaints</a>
29         </li>
30       </ul>
31     </div>
32   </div>
33
34   <main role='main' class='container'>
35     <div class='jumbotron text-center'>
36       <h1>Register a Complaint</h1>
37       <p class='lead text-info'>Click the Link Below to Register a Complaint</p>
38       <a href='upload'>This Link</a>
39     </div>
40   </main>
41 </body>
42 </html>
```

7.5 COMPLAINTS



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6   <title>Issue Tracker</title>
7   <meta name='viewport' content='width=device-width, initial-scale=1'>
8   <link rel='stylesheet' type='text/css' media='screen' href='../staticFiles/homepage.css'>
9   <script src='main.js'></script>
10 </head>
11 <body>
12   <link href='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css' rel='stylesheet' id='bootstrap-css'>
13   <script src='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js'></script>
14   <script src='//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>
15   <!-- Include the above in your HEAD tag -->
16
17   <div class='navbar navbar-expand-md navbar-dark bg-dark mb-4' role='navigation'>
18     <a class='navbar-brand' href='home'>GO BACK</a>
19     <button class='navbar-toggler' type='button' data-toggle='collapse' data-target='#navbarCollapse' aria-controls='navbarCollapse'
20       aria-expanded='false' aria-label='Toggle navigation'>
21       <span class='navbar-toggler-icon'></span>
22     </button>
23     <div class='collapse navbar-collapse' id='navbarCollapse'>
24       <ul class='navbar-nav mr-auto'>
25         <li class='nav-item active'>
26           <a class='nav-link' href='#'><span class='sr-only'>(current)</span></a>
27         <li class='nav-item'>
28           <a class='nav-link' href='#'></a>
29         </li>
30       </ul>
31     </div>
32   </div>
33
34   <main role='main' class='container'>
35     <!-- Looping -->
36     <div class='jumbotron text-center'>
37       <!-- ID: {{id[0]}} -->
38       <h1>{{title[i]}}</h1>
39       <p>{{context[i]}}</p>
40       <p>Email: {{email[i]}}</p>
41       <img style='height: 20%; width: 20%;' src='{{link[i]}}'>
42     </div>
43     <!-- End for X -->
44   </main>
45 </body>
46 </html>
```

7.6 ADMIN



```
admin.html - started - Visual Studio Code
templateFiles > admin.html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset='utf-8'>
5 <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6 <title>Issue Tracker</title>
7 <meta name='viewport' content='width=device-width, initial-scale=1'>
8 <link rel='stylesheet' type='text/css' media='screen' href='../staticFiles/homepage.css'>
9 <script src='main.js'></script>
10 </head>
11 <body>
12 <link href='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css' rel='stylesheet' id='bootstrap-css'>
13 <script src='//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js'></script>
14 <script src='//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>
15 <!-- Include the above in your HEAD tag -->
16
17 <div class='navbar navbar-expand-md navbar-dark bg-dark mb-4' role='navigation'>
18 <a class='navbar-brand' href='page'>GO BACK</a>
19 <button class='navbar-toggler' type='button' data-toggle='collapse' data-target='#navbarCollapse' aria-controls='navbarCollapse' aria-exp
20 <span class='navbar-toggler-icon'></span>
21 </button>
22 <div class='collapse navbar-collapse' id='navbarCollapse'>
23 <ul class='navbar-nav mr-auto'>
24 <li class='nav-item active'>
25 <a class='nav-link' href='#'><span class='sr-only'>(current)</span></a>
26 <li class='nav-item'>
27 </li>
28 </ul>
29 </div>
30 </div>
31
32 <main role='main' class='container'>
33 <!-- Looping -->
34 {% for i in range(0,len): %}
35 <div class='jumbotron text-center'>
36 <!-- <p>ID: {{id[i]}}</p> -->
37 <h1>{{title[i]}}</h1>
38 <p>{{context[i]}}</p>
39 <p>Email: {{email[i]}}</p>
40 <img style='height: 20%; width: 20%;' src='{{link[i]}}'>
41 </div>
42 {% endfor %}
43 </main>
44 </body>
45 </html>
```

7.7 HOMEADMIN

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6   <title>Issue Tracker</title>
7   <meta name='viewport' content='width=device-width, initial-scale=1'>
8   <link rel='stylesheet' type='text/css' media='screen' href='../staticFiles/homepage.css'>
9   <script src='main.js'></script>
10 </head>
11 <body>
12   <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">
13   <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></script>
14   <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
15   <!-- Include the above in your HEAD tag -->
16
17   <div class="navbar navbar-expand-md navbar-dark bg-dark mb-4" role="navigation">
18     <a class="navbar-brand" href="page">ISSUE TRACKER</a>
19     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-exp
20       <span class="navbar-toggler-icon"></span>
21     </button>
22     <div class="collapse navbar-collapse" id="navbarCollapse">
23       <ul class="navbar-nav mr-auto">
24         <li class="nav-item active">
25           <a class="nav-link" href="#"><span class="sr-only">(current)</span></a>
26         </li>
27         <li class="nav-item">
28           <a href="#"></a>
29         </li>
30       </ul>
31     </div>
32   </div>
33   <main role="main" class="container">
34     <div class="jumbotron text-center">
35       <h1>All Complaint</h1>
36       <p class="lead text-info">Click the Link Below to See all Complaints</p>
37       <a href="admin">This Link</a>
38     </div>
39   </main>
40 </body>
41 </html>
```

7.8 STYLE CODE

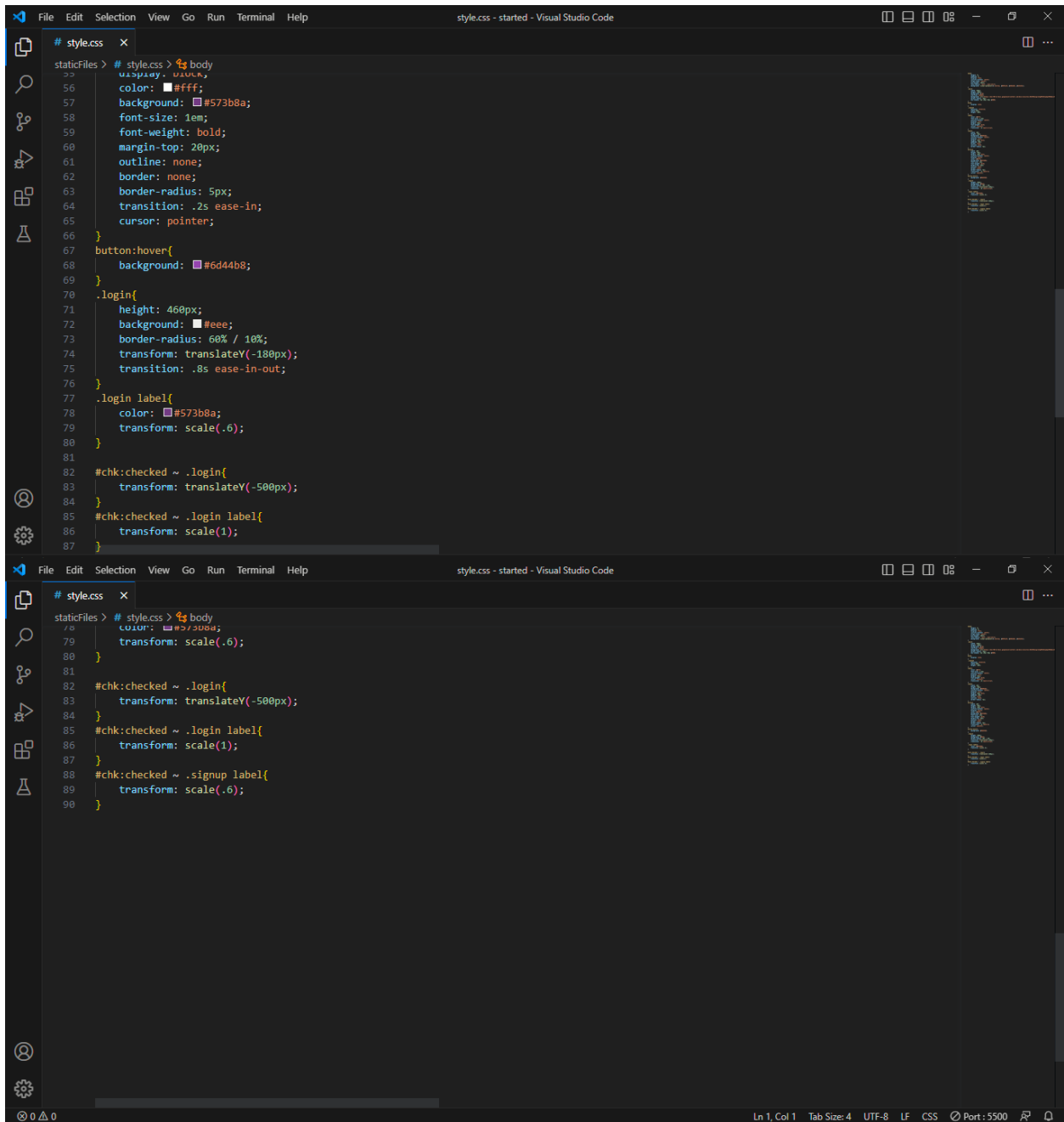
The image displays two sequential screenshots of a Visual Studio Code editor window, showing the development of a CSS file named `style.css`. The editor interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a toolbar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions, and a status bar at the bottom indicating the current line and column (Ln 1, Col 1), tab size (4), encoding (UTF-8), language (CSS), and port (5500).

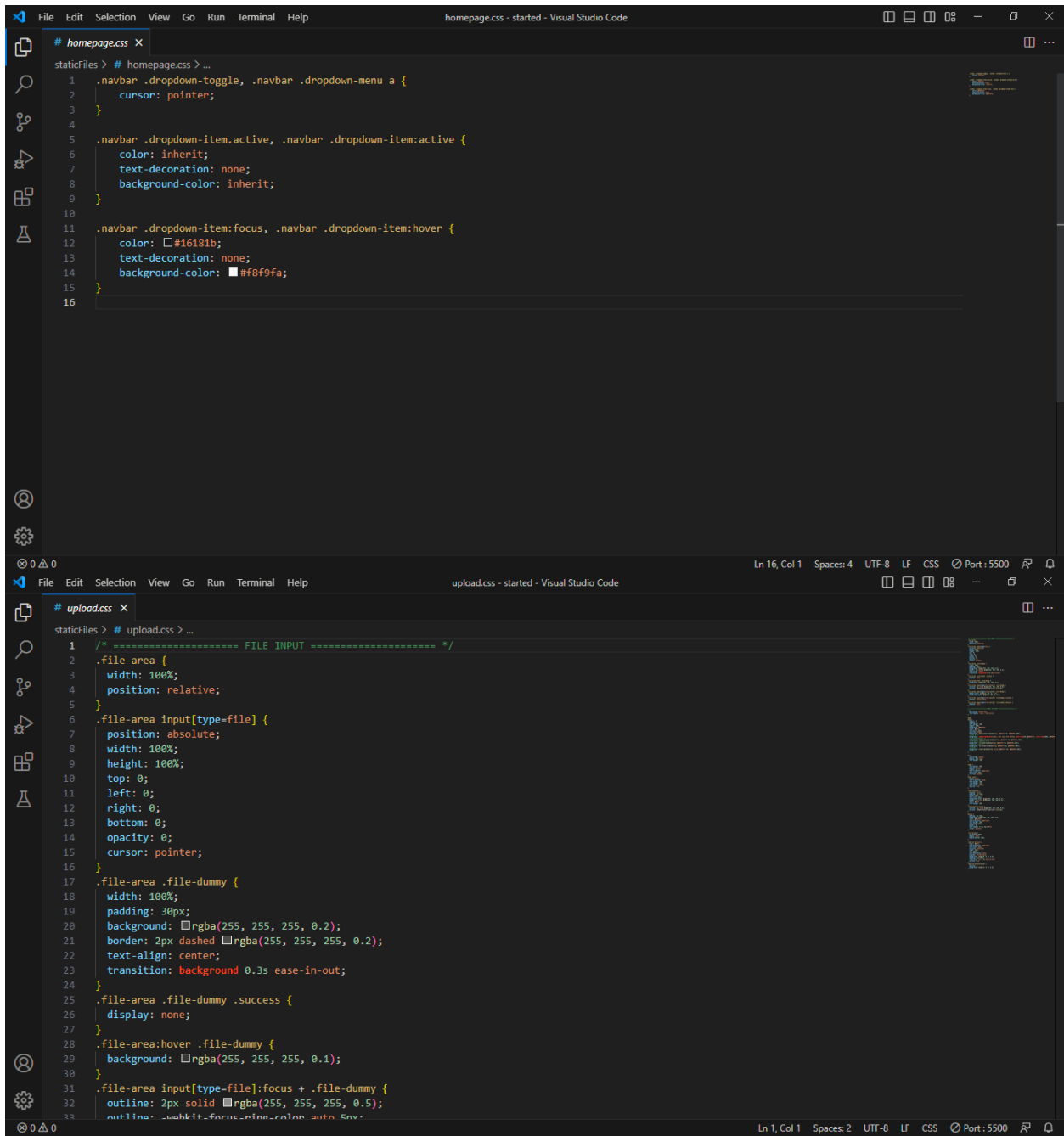
Top Screenshot: The editor shows the initial CSS code for the `body`, `.main`, `#chk`, and `.signup` selectors. The `body` selector (lines 1-10) sets the margin, padding, display, justify-content, align-items, min-height, font-family, and background. The `.main` selector (lines 11-18) sets the width, height, background, overflow, border-radius, and box-shadow. The `#chk` selector (lines 19-22) sets the display to none. The `.signup` selector (lines 23-27) sets the position, width, and height. The `label` selector (lines 28-32) sets the color, font-size, justify-content, display, and margin.

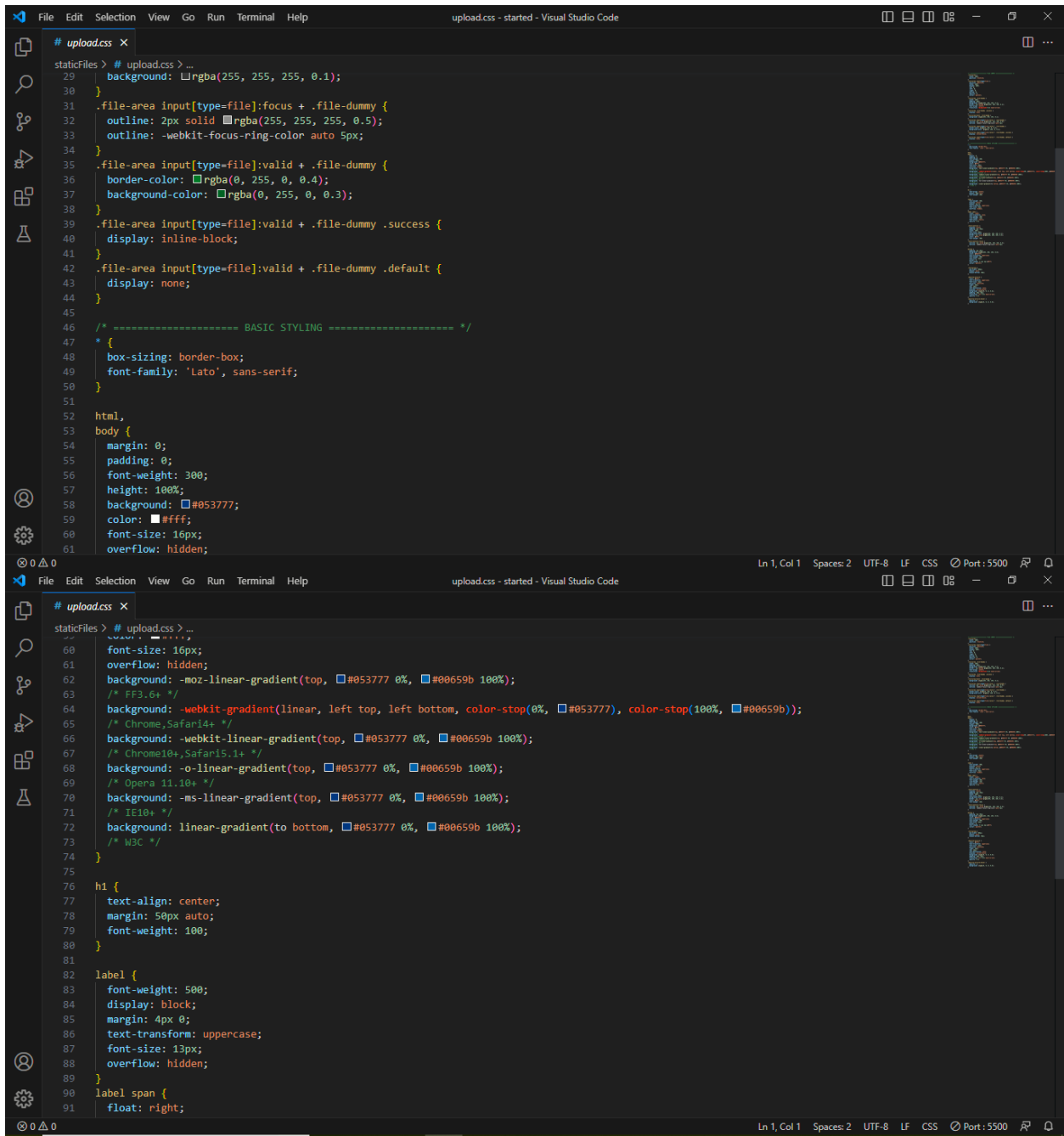
```
1 body{
2   margin: 0;
3   padding: 0;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   min-height: 100vh;
8   font-family: 'Jost', sans-serif;
9   background: linear-gradient(to bottom, #0f0c29, #302b63, #24243e);
10 }
11 .main{
12   width: 350px;
13   height: 500px;
14   background: red;
15   overflow: hidden;
16   background: url('https://doc-08-2c-docs.googleusercontent.com/docs/securesc/68c90smiglihng9534mvqm91946dmis5/fo0picsp1nhiucmc0125s29respgr4j');
17   border-radius: 10px;
18   box-shadow: 5px 20px 50px #000;
19 }
20 #chk{
21   display: none;
22 }
23 .signup{
24   position: relative;
25   width: 100%;
26   height: 100%;
27 }
28 label{
29   color: #fff;
30   font-size: 2.3em;
31   justify-content: center;
32   display: flex;
33   margin: 60px;
```

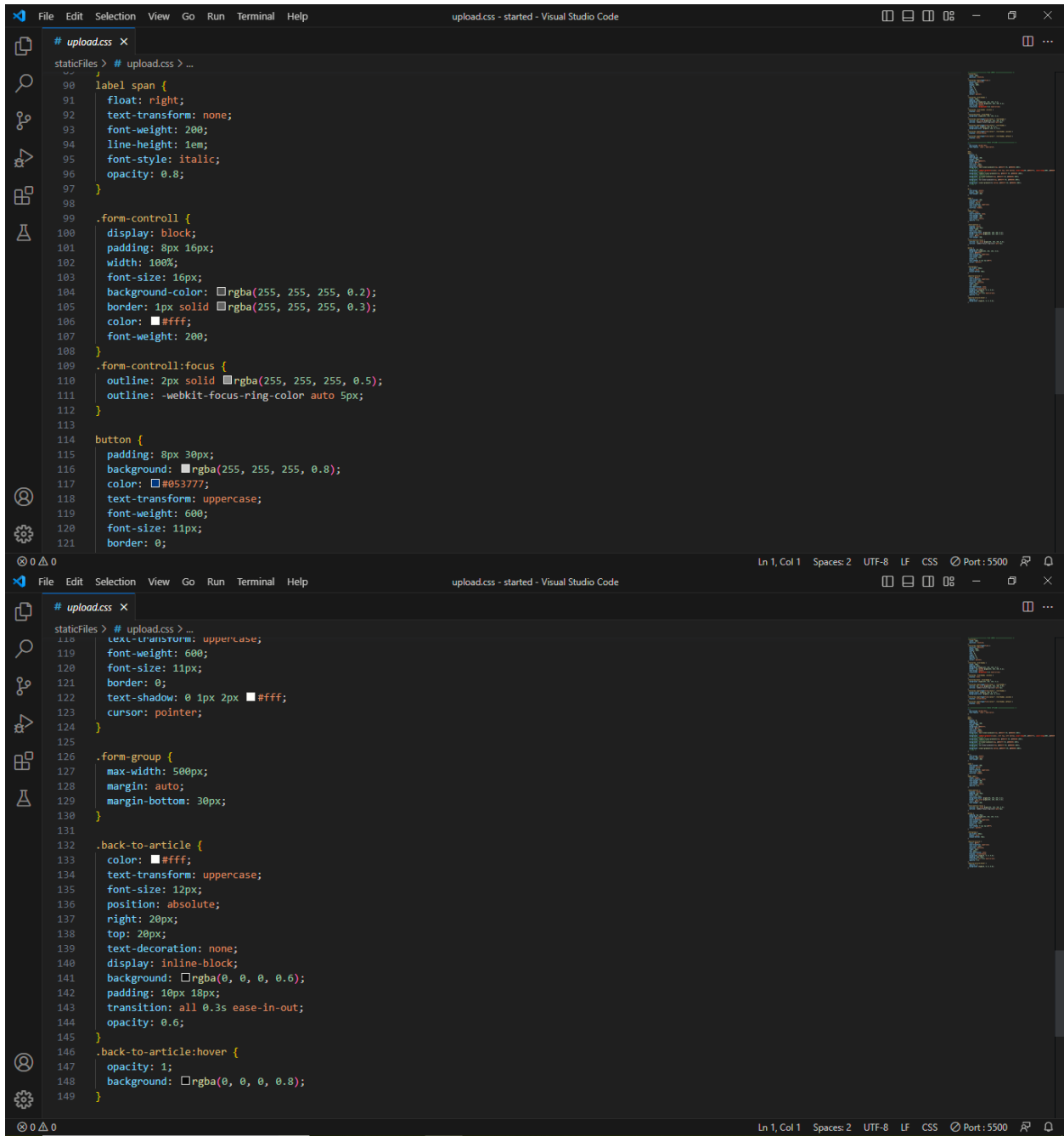
Bottom Screenshot: The editor shows the CSS code for the `input` and `button` selectors. The `input` selector (lines 38-48) sets the width, height, background, justify-content, display, margin, padding, border, outline, and border-radius. The `button` selector (lines 50-59) sets the width, height, margin, justify-content, display, color, background, font-size, and font-weight.

```
38 input{
39   width: 60%;
40   height: 20px;
41   background: #e0dede;
42   justify-content: center;
43   display: flex;
44   margin: 20px auto;
45   padding: 10px;
46   border: none;
47   outline: none;
48   border-radius: 5px;
49 }
50 button{
51   width: 60%;
52   height: 40px;
53   margin: 10px auto;
54   justify-content: center;
55   display: block;
56   color: #fff;
57   background: #573b8a;
58   font-size: 1em;
59   font-weight: bold;
60   margin-top: 20px;
```





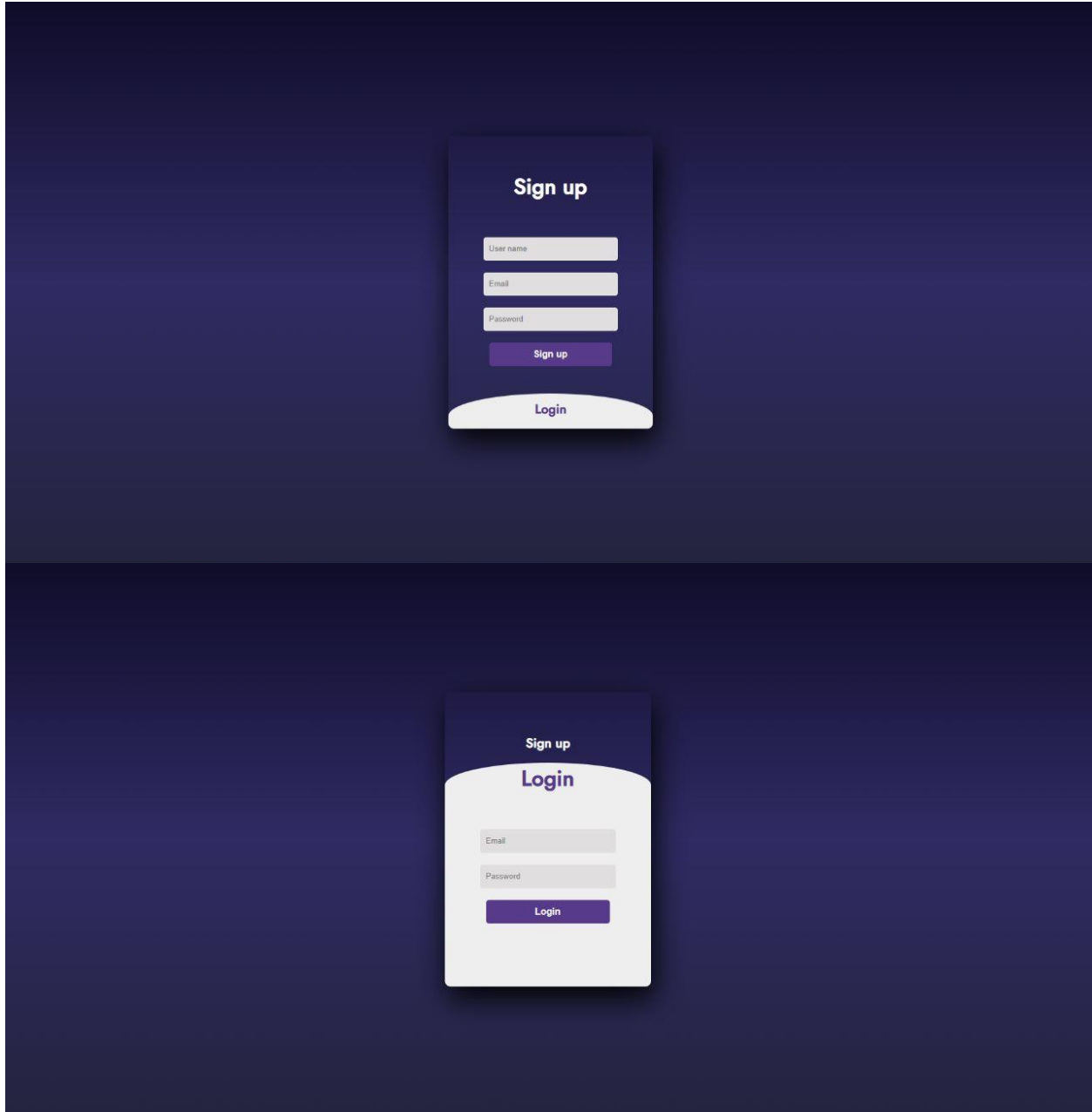


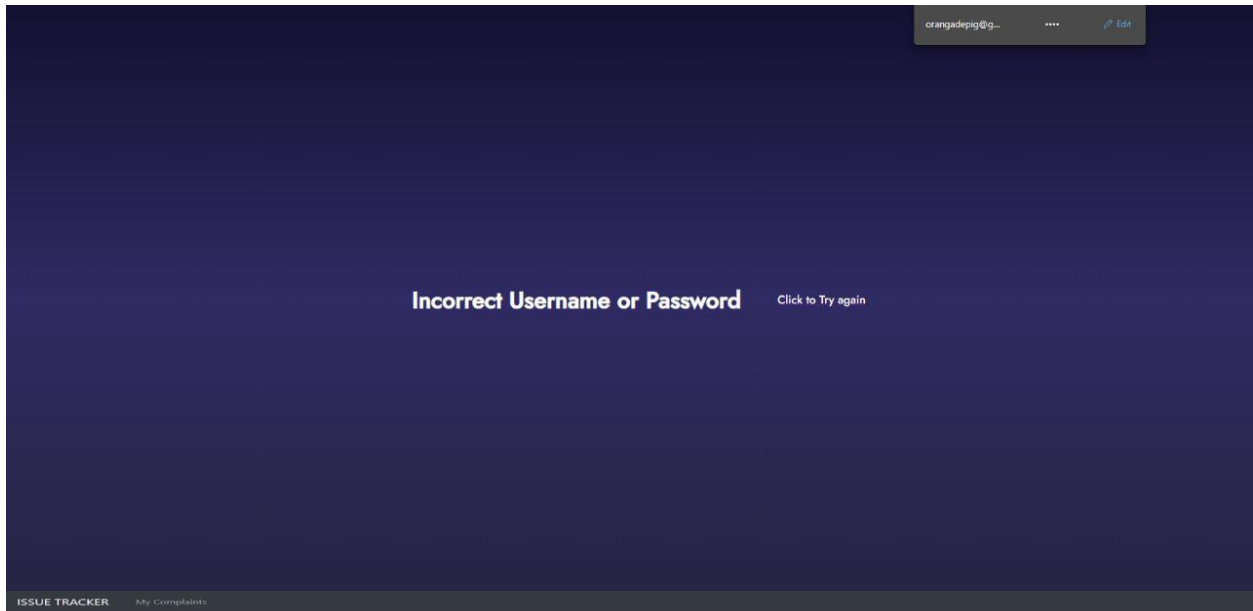


9. User Acceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done. The User Acceptance of this product is not surveyed enough to give a solid conclusion. The theoretical and hypothetical acceptance is calculated to be high enough to conclude that this product is usable and valuable.

10. OUTPUT





ISSUE REGISTER

TITLE Use title case to get a better result

CAPTION This caption should be descriptive

IMAGES Your images should be at least 400/300 v/ide

Please select some files

[SUBMIT ISSUE](#)

Water is Not Coming

In my Home (Address) No water. Please Help

Email: dekabilanmc@gmail.com



All Complaint

Click the Link Below to See all Complaints

[This Link](#)

I want to eat your Pancreas

Nice Movie

Email:



Water is Not Coming

In my Home (Address) No water. Please Help

Email: dekabilanmc@gmail.com



11. ADVANTAGES & DISADVANTAGES

> ADVANTAGES

1. Resolve issues end-to-end
2. Track issues throughout the span of their life cycles
3. Assess impact of the issues

> DISADVANTAGES

1. Leaking the personal information.
2. Fail to store the issue for his/her-self

11 CONCLUSION

We have successfully developed an application using python flask, HTML, CSS. By using the application we can find out the issues and client get the proper response.

The system successfully find out the people issues.

12 FUTURE SCOPES

Issue tracking system is not just only a central repository for tracking and resolving tickets but also provides an easy-to-access hub from effective project management. you can simply navigate through any project fill within a few clicks.

13 APPENDIX Source Code GitHub & Project Demo Link

Project Link: [Register/Login](#)

Github: [naanmudhalvan-SI/IBM--14003-1682490056: The IssueTracker: A Reliable Complaint Management System for Improved Customer Service \(github.com\)](#)

Demo: [IBM-ISSUE_TRACKER WEB APPLICATION - YouTube](#)