A

PROJECT REPORT ON

Bug Tracking System

By

AASTHA PATEL (CE001) (18CEUON120)
&
DEVANG BHALALA (CE007) (18CEUON022)
&
RAHIL BHENSDADIA (CE012) (18CEUON021)

B.Tech CE Semester-VI

Subject: Service Oriented Computing

Guided by:

Prof. Apporva Mehta	Prof. Prashant Jadav
Assistant Professor	Associate Professor
Dept. of Comp. Engg.	Dept. of Comp. Engg.

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **Service Oriented Computing** and recorded in this journal is the bonafide work of

AASTHA D. PATEL (CE001) (18CEUON120)
DEVANG P. BHALALA (CE007) (18CEUON022)
RAHIL R. BHENSDADIA (CE012) (18CEUON021)

of B.Tech semester **VI** in the branch of **Computer Engineering** during the academic year **2020-2021**.

Prof. Apoorva A. Mehta	**Prof. Prashant M. Jadav**	**Dr. C. K. Bhensdadia,**
Assistant Professor,	Associate Professor,	Head,
Dept. of Computer Engg.,	Dept. of Computer Engg.,	Dept. of Computer Engg.,
Faculty of Technology	Faculty of Technology	Faculty of Technology
Dharmsinh Desai University, Nadiad	Dharmsinh Desai University, Nadiad	Dharmsinh Desai University, Nadiad

Table of Contents

ABSTRACT

This report is about a Service Oriented Implementation of Bug Tracking System. The system provides a platform to testers and developers for rapidly generating bug alerts , monitoring the status of resolution and mentioning the steps performed for resolution of bugs by developers.

The Bug Tracking system is designed and implemented as a project for S.O.C. subject. The system mentioned in this report is implemented and designed by following the Service oriented Architecture and hence gets all the benefits of a service oriented application. There are mainly two ASP.NET Framework applications, one is the web API service application and other is the web client application. The API Service application follows REST standards and it is consumed by a web client application as a service. The logic of data handling and database connection is done by web API service and client application communicates with web API service via http request-response protocol following json message format.

INTRODUCTION

The project is a ‘Bug Tracking System’ useful for software developing companies or organizations. This is the first version of SRS for this project. The project makes the task of reporting and resolving bugs easy for the testing and development team.

The document follows the normal understandable format which can be useful for the design, planning, development and testing teams. In the beginning, the document has a brief overview and purpose of this application. Then comes the functional requirements needed to be implemented. At last, non-functional requirements which need to be followed are mentioned.

The intended audience for this document are as follows:

- System Designers
- Developers
- Managers
- Testers
- Marketing Staff
- End Users
- Documentation Writers

Tracking and managing the bugs among the teams working in a software development organization requires a lot of time and efforts when following the traditional approach. The application mentioned in this document provides a one stop solution for robust bug alert management. This application saves a lot of time and human resources compared to traditional approaches. Moreover each bug alert and the activities happening over the bug alert are stored in a persistent database this helps in effectively and reliably tracking the activities by users and tracking the work done by employees.

SOFTWARE REQUIREMENTS SPECIFICATIONS

1. Functional Requirements

1.1 Application User Management

Description: This feature includes CRUD operations for user accounts.

Priority: 1

1.1.1 Registration

Using this functionality, users such as admin, developers, system testers can register themselves to use the application.

Precondition: none.
Input: User information required for registration.
Processing: User details will be stored in the database after verification of the details.
Output: Upon successful registration, the user will be redirected to the home page of the application. Else appropriate error messages will be shown.

1.1.2 Log In

Using this functionality, already registered users can log in into the application.

Precondition: User must be already registered.
Input: User credentials to log in.
Processing: Verifying the credentials.
Output: Upon successful login, the user will be redirected to the home page of the application. Else appropriate error messages will be shown.

1.1.3 Retrieve Details of user

Using this functionality, a logged in user can see his/her profile.

Precondition: User must be already logged in.
Input: Request for details by clicking the button.
Processing: Retrieving user details from the database.
Output: Details of the requested user.

1.1.4 Update Details of the user

Using this functionality, a logged in user can update his/her details which are editable.

Precondition: User must be already logged in.
Input: Request for updating details by clicking the button and sending data.

Processing: Updating user details into the database.
Output: Updated details successful/unsuccessful message.

1.1.5 Delete user account

Using this functionality, a user account can be deleted.

Precondition: User must be already logged in.
Input: Request for deletion of account by clicking the button.
Processing: Removing the user records from the database.
Output: Account deletion successful/unsuccessful message.

1.2 Bug/Error Category Management

Description: This feature includes CRUD operations for bug categories.

Priority: 2

Precondition: Admin must be logged in.

1.2.1 Creating bug categories.

Input: Category Details required for creation.
Processing: Category details will be stored in the database.
Output: Category creation successful/unsuccessful message.

1.2.2 Retrieval of categories.

Precondition: Category must exist.

Input: Request for category details by clicking the button.
Processing: Retrieving category details from the database.
Output: Details of the requested category.

1.2.3 Edit the category.

Precondition: Category must exist.

Input: Request for updating details by clicking the button and sending data.
Processing: Updating category details into the database.
Output: Updated details successful/unsuccessful message.

1.2.4 Deletion of category.

Precondition: Category must exist.

Input: Request for deletion of category by clicking the button.
Processing: Removing the category records from the database.
Output: Category deletion successful/unsuccessful message.

1.3 Bug/Error Management

Description: This feature includes CRUD operations for bugs.

Priority: 3

Precondition: User must be logged in as Tester.

1.3.1 Creating bug entry.

Input: Bug Details required for creation.
Processing: Bug details will be stored in the database.
Output: Bug creation successful/unsuccessful message.

1.3.2 Retrieval of categories.

Precondition: Bug must exist.

Input: Request for bug details by clicking the button.
Processing: Retrieving bug details from the database.
Output: Details of the requested bug.

1.3.3 Edit the Bug.

Precondition: Bug must exist.

Input: Request for updating details by clicking the button and sending data.
Processing: Updating bug details into the database.
Output: Updated details successful/unsuccessful message.

1.3.4 Deletion of Bug.

Precondition: Bug must exist.

Input: Request for deletion of bug by clicking the button.
Processing: Removing the bug records from the database.
Output: Bug deletion successful/unsuccessful message.

1.4 Bug/Error Categorization & Assignment Management

Description: This feature includes Categorization of bugs into bug categories and assignment of developers to bugs.

Priority: 4

Precondition: There must be more than one bug category and bugs registered.

1.4.1 Bug categorization.

Admin can assign a category for each of the non-categorized bugs.

Precondition: User must be logged in as admin.

Input: Category selection for the bug.
Processing: Assigning the category to the bug in the database.
Output: Assignment confirmation message.

1.4.2 Bug assignment

Admin can assign a developer for each of the non-resolved bugs.

Precondition: User must be logged in as admin.

Input: Developer selection for the bug resolution.
Processing: Assigning the developer to the bug in the database. Mark bug status as ‘Under Resolution’.
Output: Developer assignment confirmation message.

1.4.3 Claiming Bug resolution

A developer can claim any bug for resolution.

Precondition: User must be logged in as developer.

Input: Bug selection by the developer.
Processing: Assigning the developer to the bug in the database. Mark bug status as ‘Under Resolution’.
Output: Developer assignment confirmation message.

1.5 Bug/Error Status Management

Description: This feature includes status management of bugs according to advancement of process.

Priority: 5

Precondition: User must be logged in. Bug must be assigned to the currently logged in developer.

1.5.1 Passing the responsibility of resolution

A developer can pass the responsibility of bug resolution to admin on unable to resolve for reassignment of the other developer if required.

Input: Request for reassignment by clicking the button.
Processing: Removing assignment of the developer to the bug in the database. Mark bug status as ‘One Try/ Unresolved’.
Output: Developer assignment removal confirmation message.

1.5.2 Mark status as ‘Resolved’

Input: Message for tester regarding error resolution.
Processing: Mark bug status as ‘Resolved’ and sending confirmation notification to the tester.
Output: Confirmation message.

1.5.3 Attach error report for documentation

For the documentation purpose, a developer needs to create and attach an error resolution report.

Input: Report attachment.

Processing: Store the report to the file-system and Mark the status as 'Reported'.
Output: Confirmation message.

2. Nonfunctional Requirements

2.1 Performance Requirements

The server must not be hanged in the case of too many users working concurrently on it. The delay time to load and reload the page should be as low as possible.

2.2 Safety Requirements

Concurrent manipulation of data must be managed without data corruption. A prompt should ask for the confirmation of the action.

2.3 Security Requirements

User data must be secured in the database. Unauthorized access should not be allowed. One user cannot view other user's details.

2.4 Software Quality Attributes

The project must be adaptable to most of the browsers. The application must be accessible 24*7. The application should be easy to maintain and upgrade time to time. The project should be built in such a way that its components can be reused in different sections of the project.

2.5 Business Rules

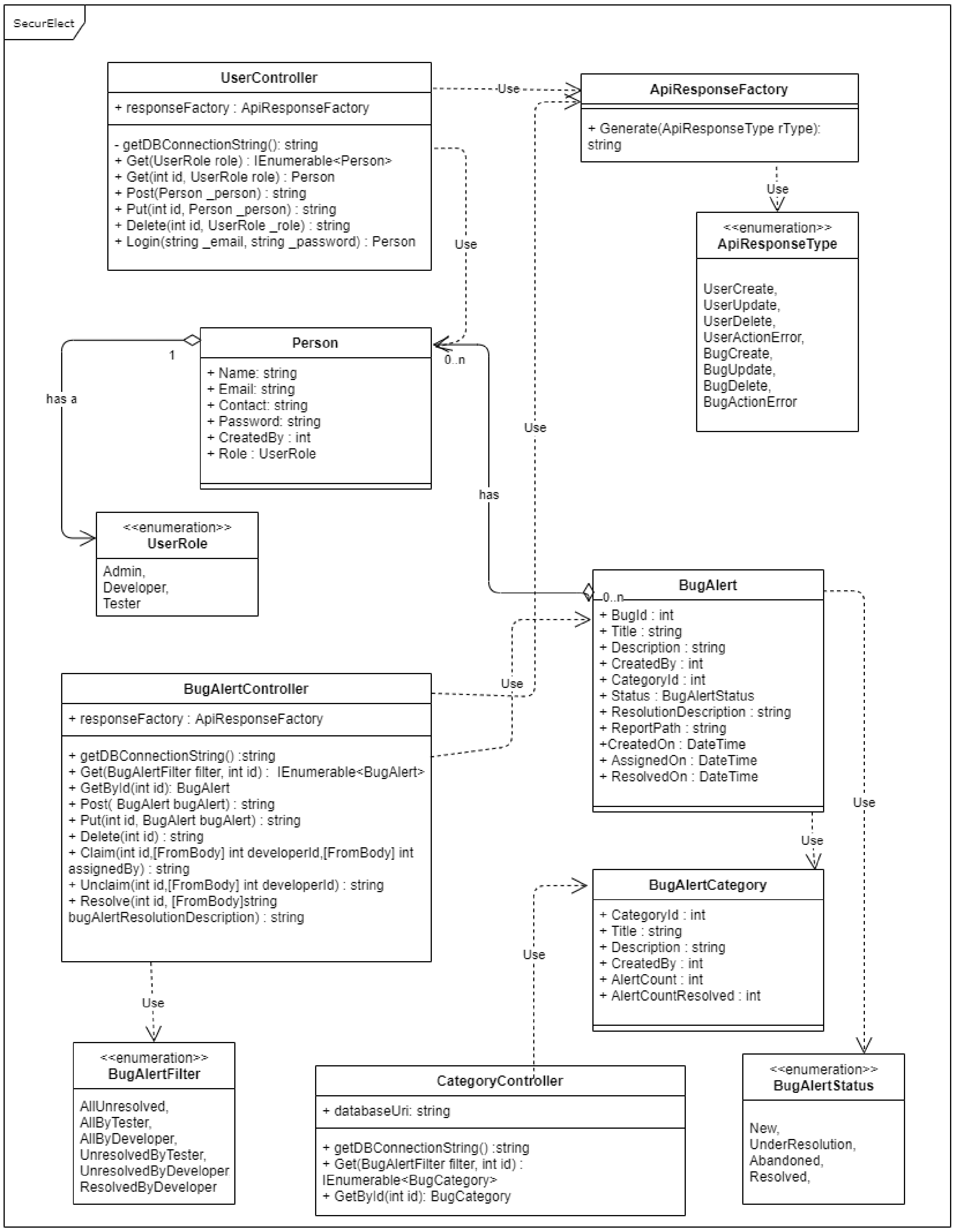
The admin has power to manage major concerns.

3. Other Requirements

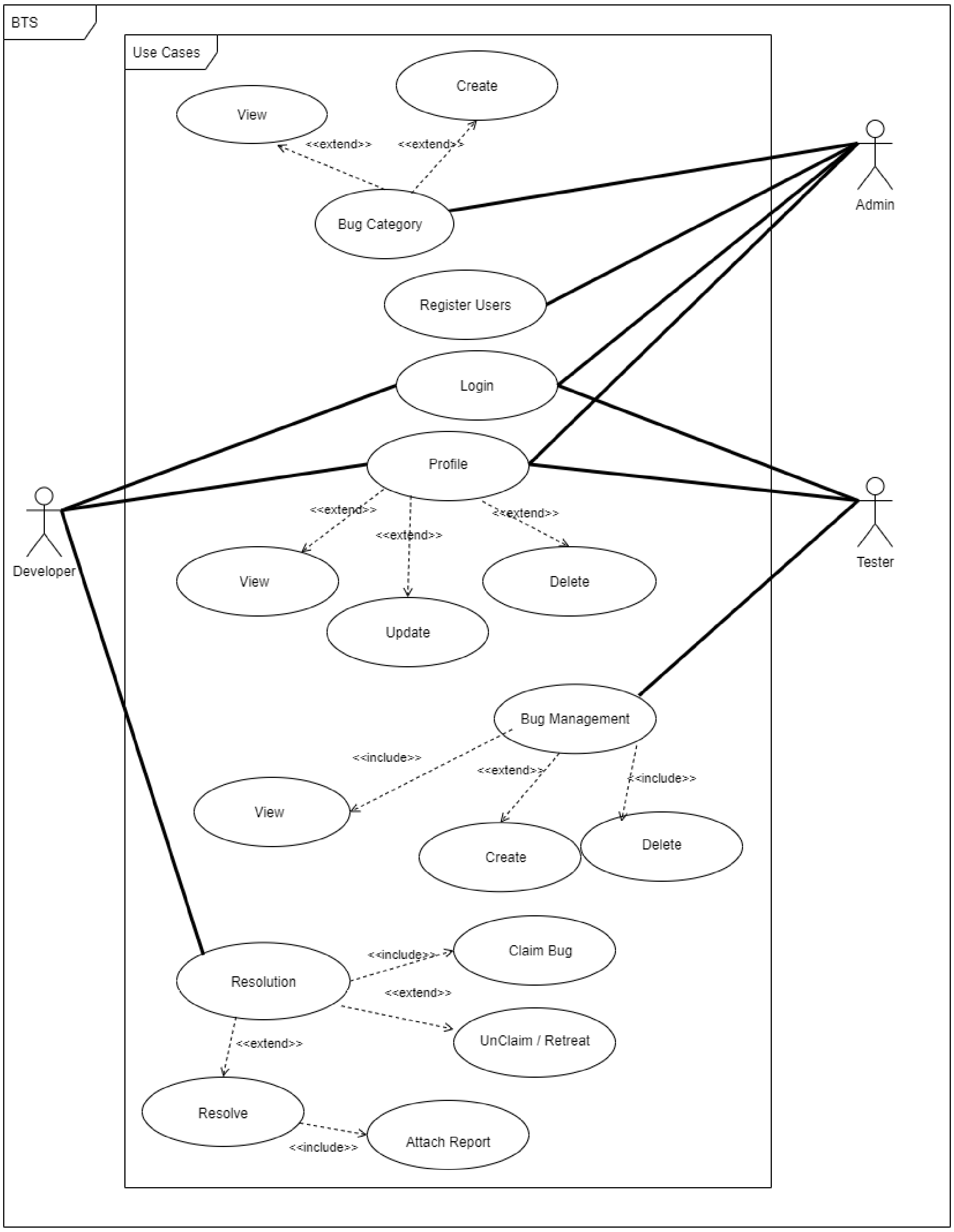
In this project, the data should be stored in the database initially. If the system permits, then the data should be stored using blockchain technology. The data should be secured and must be updated and reflected to the user immediately.

ANALYSIS AND DESIGN

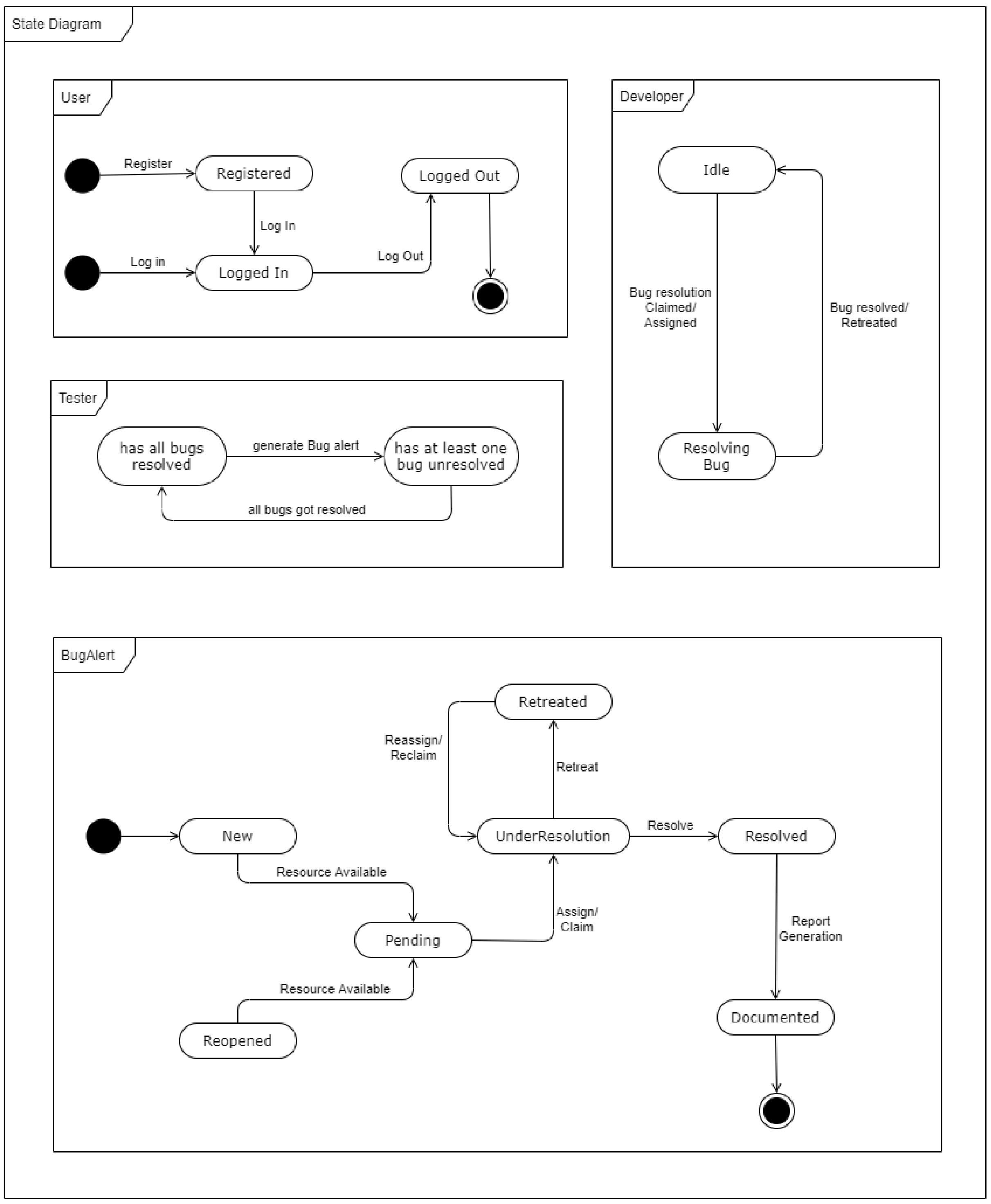
Class Diagram


SecurElect
UserController
+ responseFactory : ApiResponseFactory
- getDBConnectionString(): string
+ Get(UserRole role) : IEnumerable<Person>
+ Get(int id, UserRole role) : Person
+ Post(Person _person) : string
+ Put(int id, Person _person) : string
+ Delete(int id, UserRole _role) : string
+ Login(string _email, string _password) : Person
Use
ApiResponseFactory
+ Generate(ApiResponseType rType): string
Use
<<enumeration>>
ApiResponseType
UserCreate,
UserUpdate,
UserDelete,
UserActionError,
BugCreate,
BugUpdate,
BugDelete,
BugActionError
Use
Person
+ Name: string
+ Email: string
+ Contact: string
+ Password: string
+ CreatedBy : int
+ Role : UserRole
1
has a
0..n
Use
has
<<enumeration>>
UserRole
Admin,
Developer,
Tester
BugAlert
0..n
+ BugId : int
+ Title : string
+ Description : string
+ CreatedBy : int
+ CategoryId : int
+ Status : BugAlertStatus
+ ResolutionDescription : string
+ ReportPath : string
+CreatedOn : DateTime
+ AssignedOn : DateTime
+ ResolvedOn : DateTime
Use
BugAlertController
+ responseFactory : ApiResponseFactory
+ getDBConnectionString() :string
+ Get(BugAlertFilter filter, int id) : IEnumerable<BugAlert>
+ GetById(int id): BugAlert
+ Post(BugAlert bugAlert) : string
+ Put(int id, BugAlert bugAlert) : string
+ Delete(int id) : string
+ Claim(int id,[FromBody] int developerId,[FromBody] int assignedBy) : string
+ Unclaim(int id,[FromBody] int developerId) : string
+ Resolve(int id, [FromBody]string bugAlertResolutionDescription) : string
Use
Use
BugAlertCategory
+ CategoryId : int
+ Title : string
+ Description : string
+ CreatedBy : int
+ AlertCount : int
+ AlertCountResolved : int
Use
Use
<<enumeration>>
BugAlertFilter
AllUnresolved,
AllByTester,
AllByDeveloper,
UnresolvedByTester,
UnresolvedByDeveloper
ResolvedByDeveloper
CategoryController
+ databaseUri: string
+ getDBConnectionString() :string
+ Get(BugAlertFilter filter, int id) : IEnumerable<BugCategory>
+ GetById(int id): BugCategory
<<enumeration>>
BugAlertStatus
New,
UnderResolution,
Abandoned,
Resolved,

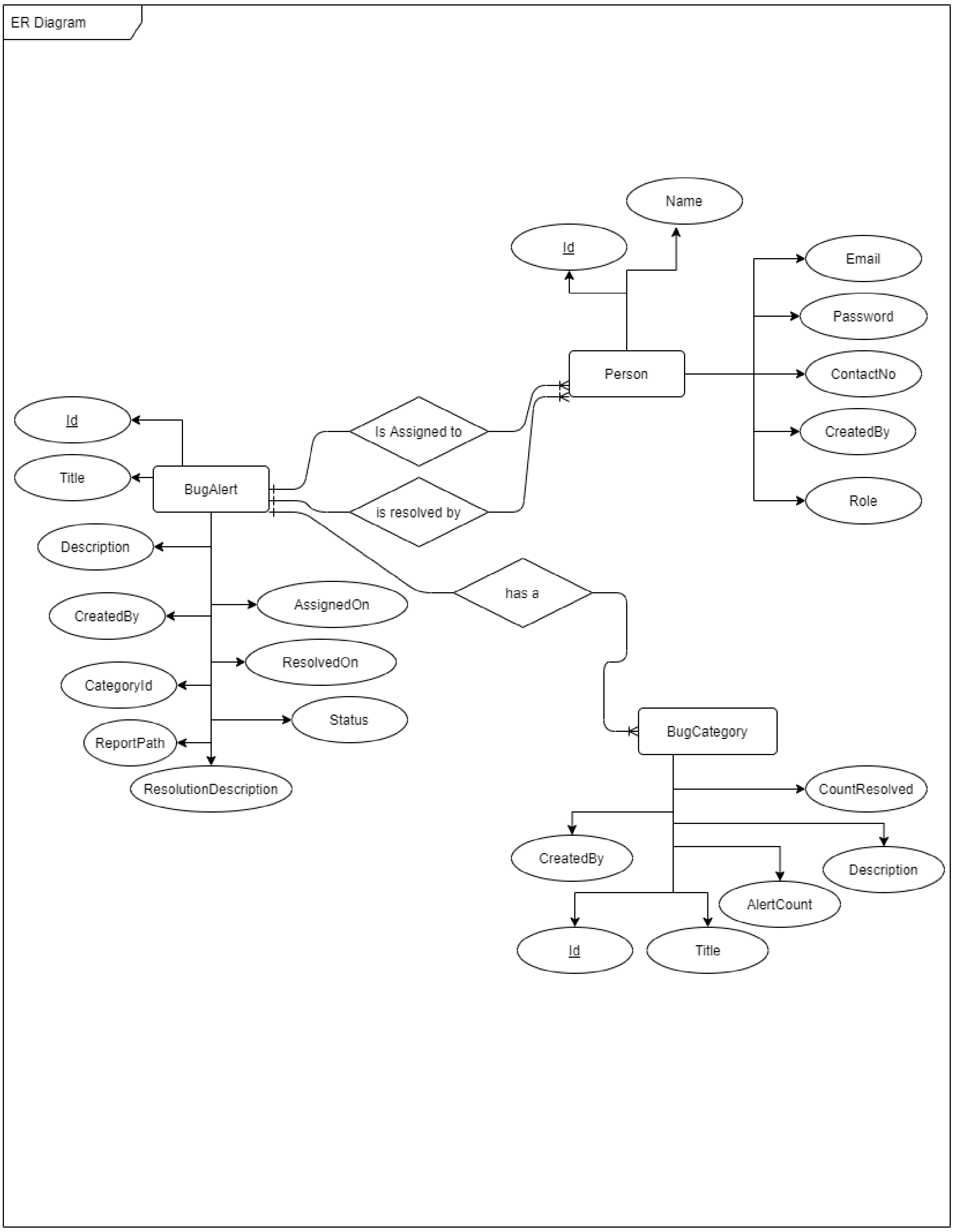
Use Case Diagram



State Diagram



E-R Diagram


ER Diagram
Name
Id
Email
Password
Person
ContactNo
CreatedBy
Role
Id
Is Assigned to
Title
BugAlert
is resolved by
Description
has a
CreatedBy
AssignedOn
ResolvedOn
CategoryId
Status
ReportPath
BugCategory
ResolutionDescription
CountResolved
CreatedBy
Description
AlertCount
Id
Title

Data Dictionary

1. Person

Field Name	DataType	Field Length	Description
id	int	-	Unique Id generated by database
Name	string	250	User Name
email	string	250	User’s email address
ContactNo	string	400	User’s contact number
Role	string	256	Role of user
Password	String	300	Password for user authentication
CreatedBy	int	-	Id of person who has created this person entry

2. Bug Alert

Field Name	DataType	Field Length	Description
id	int	-	Unique Id generated by database
Title	string	50	Title for the bug alert
Description	string	500	Description for mentioning the details of the Bug
CreatedBy	int	-	Id of the person who created this alert
CategoryId	int	-	Id of bug category
Status	string	50	Status of the bugAlert
ResolutionDescription	String	500	Description of steps taken to resolve the error
ReportFilePath	string	-	Path of detailed resolution report if uploaded
CreatedOn	DateTime	-	Date when the bugAlert is created
AssignedOn	DateTime	-	Date when the BugAlert is assigned to any developer for resolution
ResolvedOn	DateTime	-	Date when the BugAlert is resolved

3. BugAssignment

Field Name	DataType	Field Length	Description
BugAlertid	int	-	ForeignKey of BugAlert Table
DeveloperId	int	-	ForeignKey of Person Table
AssignedBy	int	-	ForeignKey of Person Table

4. BugCategory

Field Name	DataType	Field Length	Description
id	int	-	Unique Id generated by database
Title	string	50	Title for the bug alert
Description	string	500	Description for mentioning the details of the Bug
CreatedBy	int	-	Id of the person who created this alert
CreatedBy	int	-	Id of creator of category, foreignKey of person table
AlertCount	int	-	Count Of alerts created of this category
AlertResolvedCount	int	-	Count of alerts which are of this category and resolved

IMPLEMENTATION DETAILS

1) Modules created and their brief description

a) Auth Module
   - This module takes care of authentication of the user by validating the email and password for an account.

b) User Module
   - This module contains functionalities regarding user profile such as user registration, view profile details, update profile and delete the account.

c) Bug Alert Module
   - This module focuses on the features concerning the bug alerts. Features like creating, retrieving, updating and deleting bug alerts are implemented in this module. A feature that can get the bug list according to the filters a user wants is also implemented in this module.

d) Bug Status Change Module
   - This module provides a facility to change the status of the bug alert that a bug alert can have.

2) Function prototypes which implements major functionality.

a) Auth Controller
   - public IHttpActionResult Login([FromBody] AuthModel authModel)

b) User Controller
   - public IHttpActionResult Get([FromUri]UserRole role)
   - public IHttpActionResult Get(string id, [FromUri]UserRole role=UserRole.Any)
   - public HttpResponseMessage Post([FromBody]Person _person)
   - public HttpResponseMessage Put(int id, [FromBody]Person _person)
   - public HttpResponseMessage Delete(int id, [FromUri]UserRole _role=UserRole.Any)

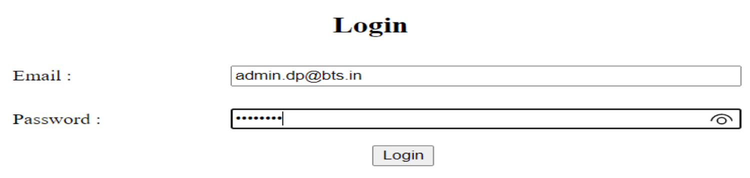
c) Bug Alert Controller
   - public IHttpActionResult GetBugList([FromUri]BugAlertFilter filter, [FromUri]int personId)
   - public IHttpActionResult GetBug(int id)
   - public HttpResponseMessage Post([FromBody] BugAlert bugAlert)

- ■ public HttpResponseMessage Put(int id, [FromBody]BugAlert bugAlert)
- ■ public HttpResponseMessage Delete(int id)

d) Bug Alert Status Change Controller

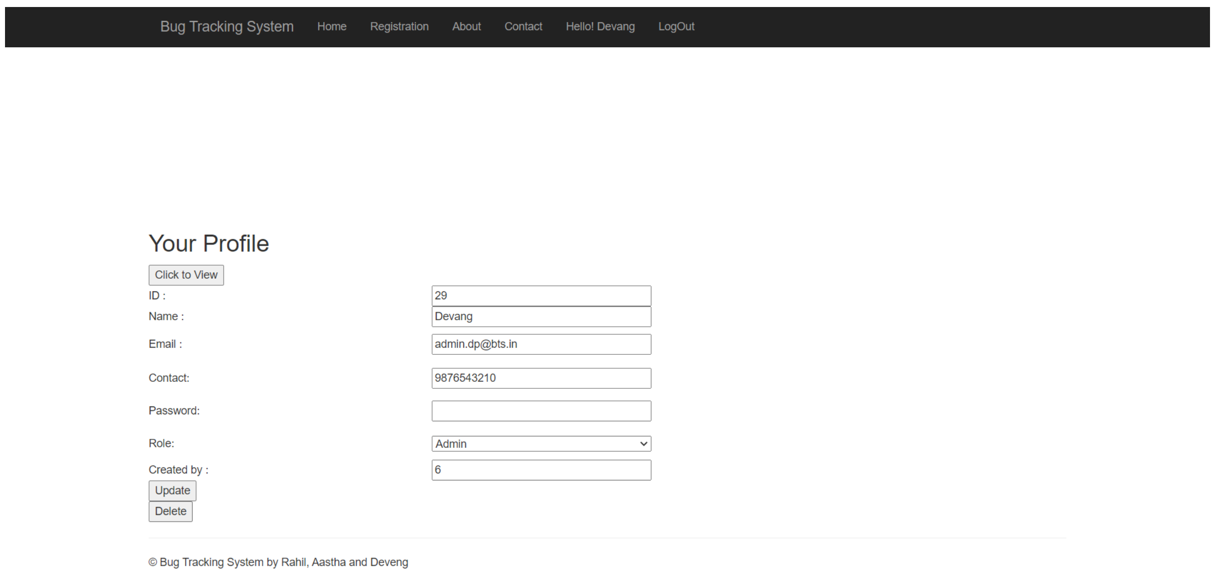
- ■ public HttpResponseMessage Claim([FromBody] StatusChangeModel statusChangeModel)
- ■ public HttpResponseMessage Retreat([FromBody] StatusChangeModel statusChangeModel)
- ■ public HttpResponseMessage Resolve([FromBody] StatusChangeModel statusChangeModel)

SCREENSHOTS



Login

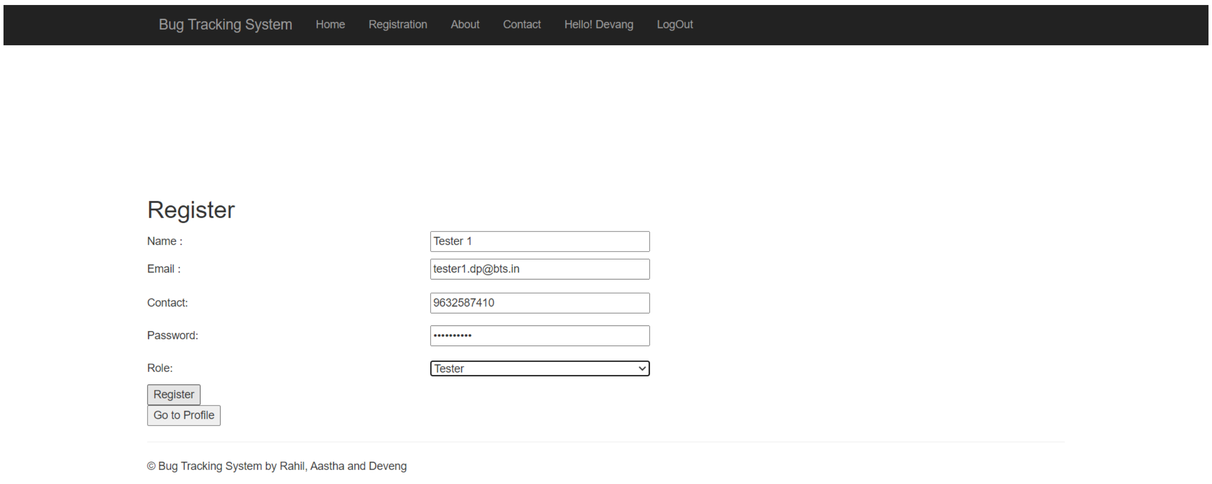
Email :	admin.dp@bts.in
Password :	••••••••



Your Profile

ID :	29
Name :	Devang
Email :	admin.dp@bts.in
Contact:	9876543210
Password:	
Role:	Admin
Created by :	6

© Bug Tracking System by Rahil, Aastha and Deveng

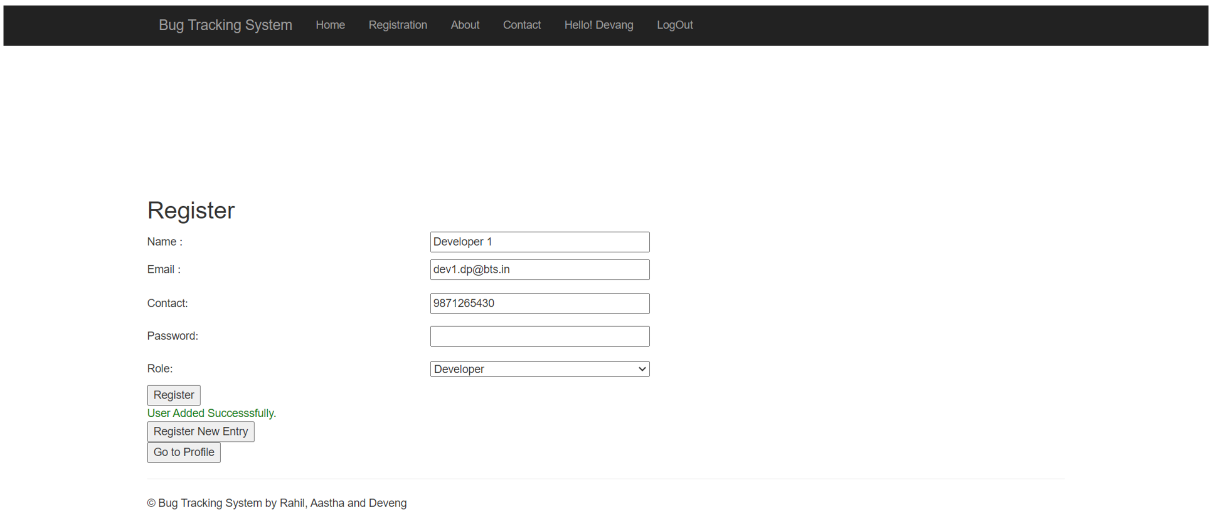

Bug Tracking System Home Registration About Contact Hello! Devang LogOut

Register

Name :	Tester 1
Email :	tester1.dp@bts.in
Contact:	9632587410
Password:	••••••••••
Role:	Tester

Register
Go to Profile

© Bug Tracking System by Rahil, Aastha and Deveng

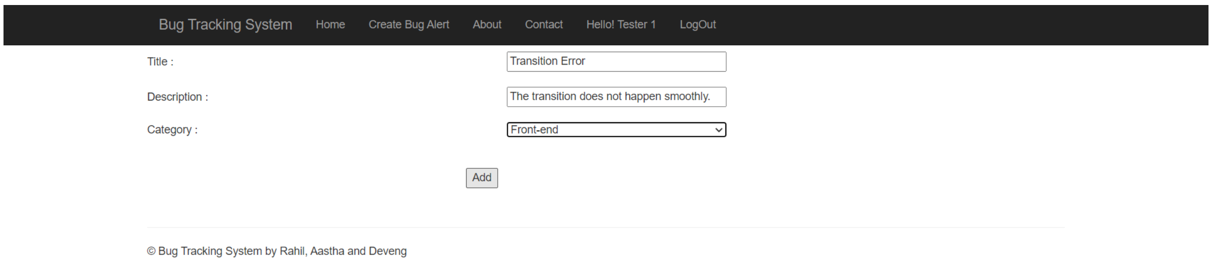

Bug Tracking System Home Registration About Contact Hello! Devang LogOut

Register

Name :	Developer 1
Email :	dev1.dp@bts.in
Contact:	9871265430
Password:	
Role:	Developer

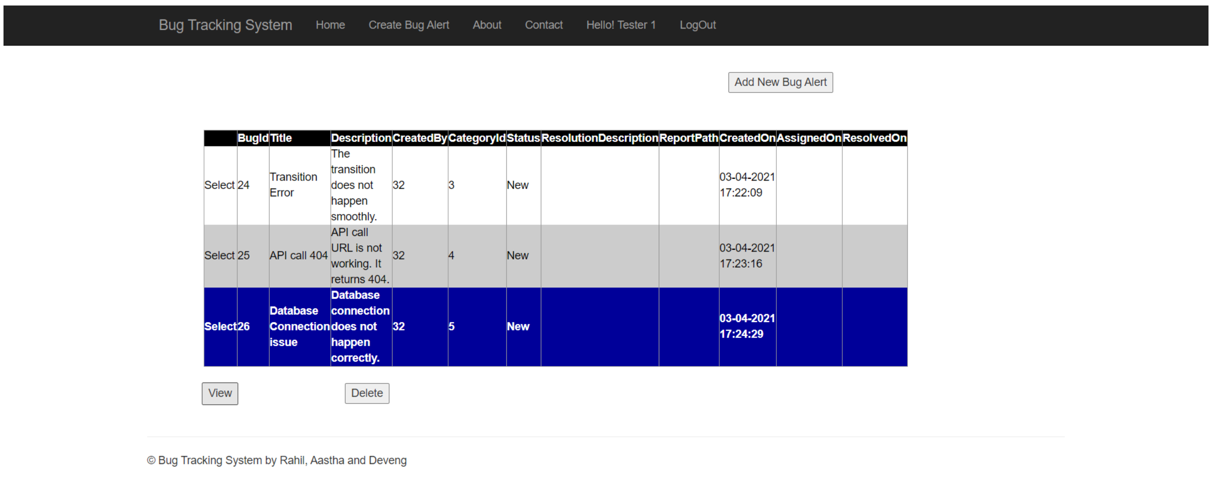
Register
User Added Successfully.
Register New Entry
Go to Profile

© Bug Tracking System by Rahil, Aastha and Deveng



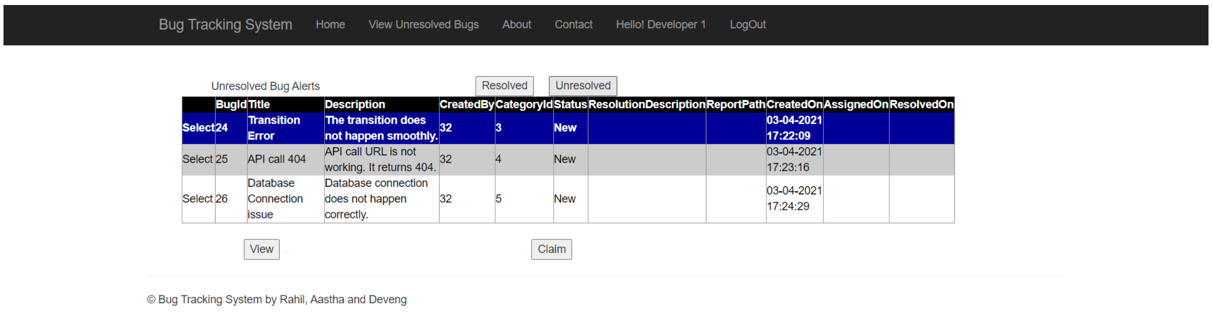
Title :	Transition Error
Description :	The transition does not happen smoothly.
Category :	Front-end

© Bug Tracking System by Rahil, Aastha and Deveng



	BugId	Title	Description	CreatedBy	CategoryId	Status	ResolutionDescription	ReportPath	CreatedOn	AssignedOn	ResolvedOn
Select	24	Transition Error	The transition does not happen smoothly.	32	3	New			03-04-2021 17:22:09		
Select	25	API call 404	API call URL is not working. It returns 404.	32	4	New			03-04-2021 17:23:16		
Select	**26**	**Database Connection issue**	**Database connection does not happen correctly.**	**32**	**5**	**New**			**03-04-2021 17:24:29**		

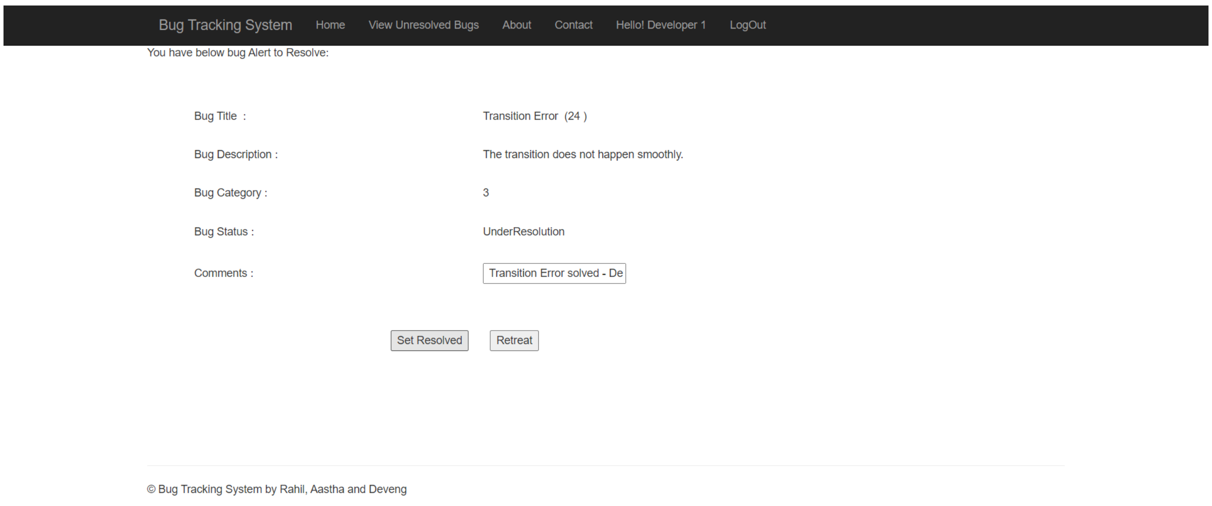
© Bug Tracking System by Rahil, Aastha and Deveng



Unresolved Bug Alerts

	BugId	Title	Description	CreatedBy	CategoryId	Status	ResolutionDescription	ReportPath	CreatedOn	AssignedOn	ResolvedOn
Select	24	Transition Error	The transition does not happen smoothly.	32	3	New			03-04-2021 17:22:09		
Select	25	API call 404	API call URL is not working. It returns 404.	32	4	New			03-04-2021 17:23:16		
Select	26	Database Connection issue	Database connection does not happen correctly.	32	5	New			03-04-2021 17:24:29		

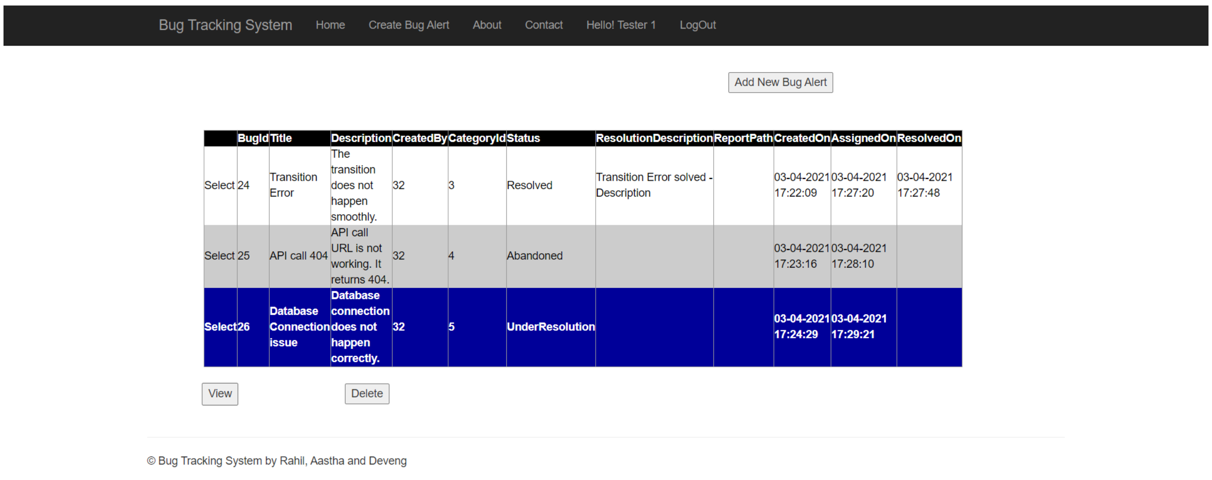
© Bug Tracking System by Rahil, Aastha and Deveng



You have below bug Alert to Resolve:

Bug Title :	Transition Error (24)
Bug Description :	The transition does not happen smoothly.
Bug Category :	3
Bug Status :	UnderResolution
Comments :	Transition Error solved - De

© Bug Tracking System by Rahil, Aastha and Deveng

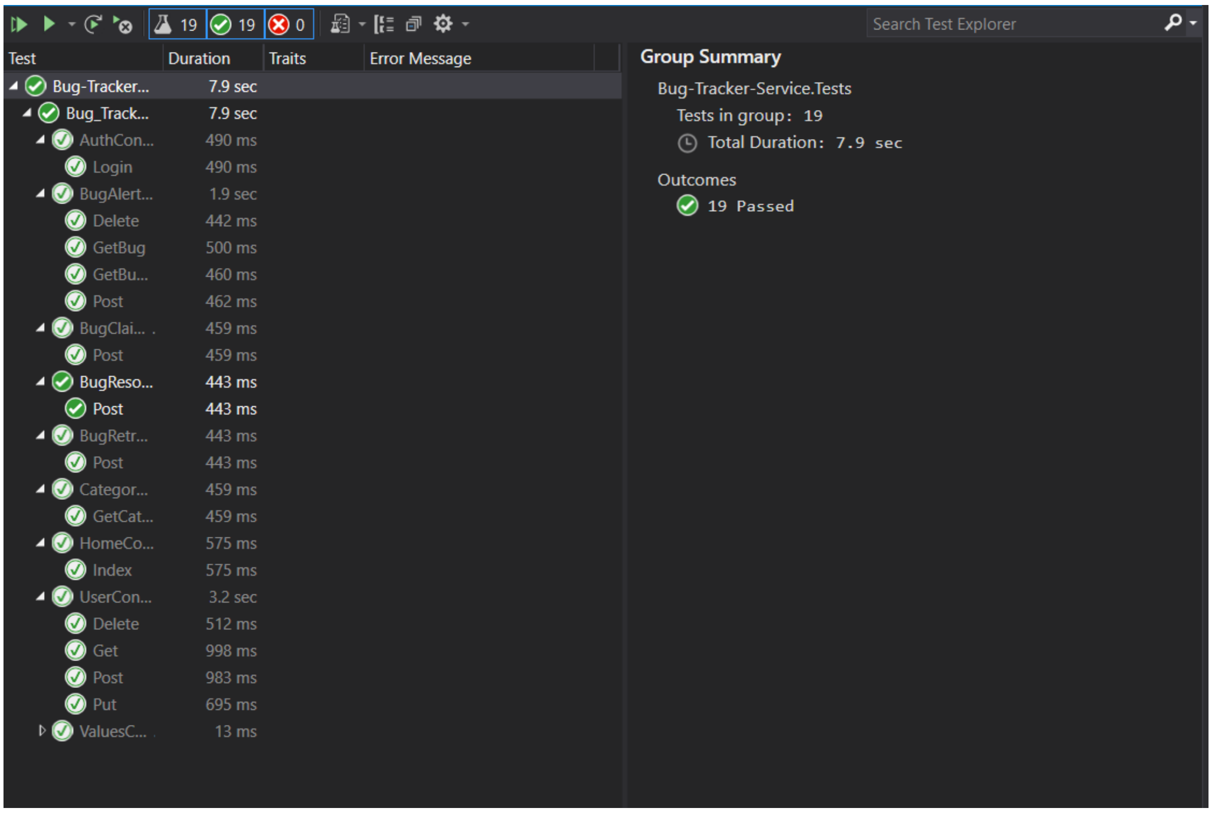


	BugId	Title	Description	CreatedBy	CategoryId	Status	ResolutionDescription	ReportPath	CreatedOn	AssignedOn	ResolvedOn
Select	24	Transition Error	The transition does not happen smoothly.	32	3	Resolved	Transition Error solved - Description		03-04-2021 17:22:09	03-04-2021 17:27:20	03-04-2021 17:27:48
Select	25	API call 404	API call URL is not working. It returns 404.	32	4	Abandoned			03-04-2021 17:23:16	03-04-2021 17:28:10	
Select	**26**	**Database Connection issue**	**Database connection does not happen correctly.**	**32**	**5**	**UnderResolution**			**03-04-2021 17:24:29**	**03-04-2021 17:29:21**	

© Bug Tracking System by Rahil, Aastha and Deveng

TESTING

Testing of each actions are done using UnitTesting project created specially for .NET Framework WebApi Project



```
19   19   0                                  Search Test Explorer

Test               Duration   Traits   Error Message     Group Summary
Bug-Tracker...     7.9 sec                                 Bug-Tracker-Service.Tests
  Bug_Track...     7.9 sec                                   Tests in group: 19
    AuthCon...     490 ms                                    Total Duration: 7.9 sec
      Login        490 ms
    BugAlert...    1.9 sec                                 Outcomes
      Delete       442 ms                                    19 Passed
      GetBug       500 ms
      GetBu...     460 ms
      Post         462 ms
    BugClai...     459 ms
      Post         459 ms
    BugReso...     443 ms
      Post         443 ms
    BugRetr...     443 ms
      Post         443 ms
    Categor...     459 ms
      GetCat...    459 ms
    HomeCo...      575 ms
      Index        575 ms
    UserCon...     3.2 sec
      Delete       512 ms
      Get          998 ms
      Post         983 ms
      Put          695 ms
    ValuesC...     13 ms
```

CONCLUSION AND FUTURE WORK

As the goal of this project was to implement the system according to Service oriented Architecture. The goal has been achieved and a complete cycle of bug tracing is managed by the application.

Moreover this system can be further extended by adding email notifications and providing code & branch references from version control systems like github. Features like rewarding score for generating and resolving bugs and creating a leaderboard for employees to develop a healthy competitive environment among employees can also be developed.

REFERENCES

- Web API In ASP.NET
- ASP.NET Web APIs | Rest API's with .NET and C#
- ASP.NET Web API Tutorials
- ASP.NET Web API Tutorial | Web API Tutorial
- ASP.NET Web API Tutorials For Begineers and Professionals
- Web API Tutorial - JavaTpoint
- Unit Testing Controllers in ASP.NET Web API 2
- Action Results in Web API 2 - ASP.NET 4.x