Bug Tracking System
BugVITa
Software Requirements Specification

Version: 1.0

Signatures

Date	Revision	Approved By
2021.03.23	1.0	Aashish Sharma
2021.03.23	1.0	Priyanshu Baban Gaikwad
2021.03.23	1.0	Srikanth Balakrishna

List of Contributors

Name	Initials	Email
Aashish Sharma	AS	aashish.sharma2019@vitstudent.ac.in
Priyanshu Gaikwad	PG	priyanshu.gaikwad2019@vitstudent.ac.in
Srikanth Balakrishna	SB	srikanth.balakrishna2019@vitstudent.ac.in

Change History

Revision	Date	Description
1.0	2021.03.23	Initial Revision

Preface

This document represents the Software Requirements Specification for the BugVITa project. The document begins with an Introduction section that describes the purpose of the document and what is considered to be in the scope of this document as well as what is outside the scope of this document.

The next section is an Overall Description of the requirements and functions. This section includes the overall constraints that the project is working within as well as the assumptions made by the project as far as the defining the requirements is concerned. Lastly, the project dependencies are also listed in this section.

The Specific Requirements section comes next and is the most important section of this document. This section goes into detail about each specific requirement of BugVITa. A description, use case with sequence of events, and any related requirements is given for each requirement. This section also gives a detailed description of the External Interfaces for the project including a description of the user interface for the software.

The Specific Requirements section also describes the Performance Requirements that are to met by BugVITa. Design Constraints and Standards Compliance are also considered in this section. Lastly, various System Attributes are discussed including Maintainability, Security, and Portability.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document describes the software requirements and specification for the system in the user and system level, detailed functional requirement are mentioned in the document. The requirement will be illustrated and presented with the help of diagrams which are used to show complicated interactions.

This document also provides a description of any project dependencies that need to be explicitly expressed. Along with the requirements descriptions, it is also the purpose of this document to describe any performance requirements that need to be met. If there are any standards that need to be considered when developing the software are also listed.

Lastly, the purpose of this document is to communicate the system attributes of the BugVITa software. These system attributes include reliability, availability, scalability, maintainability, and portability.

1.2 Scope

It is within the scope of the Software Requirements Specification to describe the specific system requirements of BugVITa.
Bug and issue tracking systems are often implemented as a part of integrated project. Some bug trackers are designed to be used with the distributed revision control software. These distributed bug trackers allow bug reports to be conveniently read, added to the database or updated while a developer is offline. Recently, commercial bug tracking system have also begun to integrate with distributed revision control. All type of bug tracking systems are conventionally viewed as a distinct type of software, so we will develop a bug/issue tracking software for all types of bugs. So, in this document we will provide information regarding BugVITa’s system attributes,requirements,standards etc.

It is outside the scope of this document to specify what types are bugs are going to be involved,or how they will be solved, and how certain backend implementation may happen.
It is also outside the scope of this document to describe in any detail at all how certain mentioned standards or technologies work and operate.

1.3 Definitions, Acronyms, and Abbreviations

Table of Definitions, Acronyms, and Abbreviations

Definition, Acronym, or Abbreviation	**Description**
Bug	In the computer world, a bug is an error in a software program. It may cause a program to crash or show undesired events that results in the problem suffered by the users when they use the software.
BTS	Bug Tracking System
SRS	Software Requirements Specification.

UC	Use case
DFD	Data flow diagram
STD	State Transition diagram
ERD	Entity Relationship Diagram

1.4 References

Table of References

References	Description
BugZilla	.A PERL coded application that was used to track bugs for Mozilla.

2 Overall Description

2.1 Product Perspective

Bug and issue tracking systems are often implemented as a part of integrated project. Some bug trackers are designed to be used with the distributed revision control software. These distributed bug trackers allow bug reports to be conveniently read, added to the database or updated while a developer is offline. Recently, commercial bug tracking system have also begun to integrate with distributed revision control. All type of bug tracking systems are conventionally viewed as a distinct type of software, so we will develop a bug/issue tracking software for all types of bugs. So, in this document we will provide information regarding BugVITa's system attributes,requirements,standards etc.

2.2 Product Functions

The following is a table of the requirements that the system SHALL meet.
Requirements originated from the Internal StakeHolders.

Table of Shall Requirements

ID	Shall Requirement
	System shall allow all users to 'login' in order to work on a project.
	All users shall be able to add a bug report by filling all the necessary details required.
	All users shall have the ability to search for a bug from the database by name
	All users shall be able to search for a bug that created within a particular time frame ,i.e, between Dates X and Y(e.g: Between 3rd and 5th January)
	All users shall be able to search for a bug using any particular attribute of the bug (e.g: Search all bugs that have Status=resolved)
	The author of a certain bug shall be allowed to modify its details.
	The author of a bug shall be allowed to delete it as well
	The admin/developer shall be able to add a bug report by filling all the necessary details required.
	The admin/developer shall be able to modify any bug.
	The admin/developer shall be able to mark any bug as resolved.
	The admin/developer shall be able to mark a bug as duplicate. The duplicate bug will then redirect to the original bug.

2.3 Constraints

The follow is a table of the design constraints that the system SHALL meet.
The list of constraints was produced by the Internal StakeHolders.

Table of Design Constraints

ID	***Shall Requirement***
1	The user interface shall also be user friendly so that everyone can use it.
2	Protection at Admin level,system shall try and have security for unauthorized access of data by using a password for database access.
3	Content of the Bug Report shall not be restricted by the system.
4	Honesty of the Bug reporter(author) shall not be the responsibility of the system.
5	Tester shall only be able to add,view,search a bug as well as modify only limited part of his own reports only while the bug is Open.
6	Administrator should be able to link a particular registered user to a desired product/project.
7	It shall not be the responsibility of the system to deal with other aspects of project management such as version control,etc :

2.4 User Characteristics

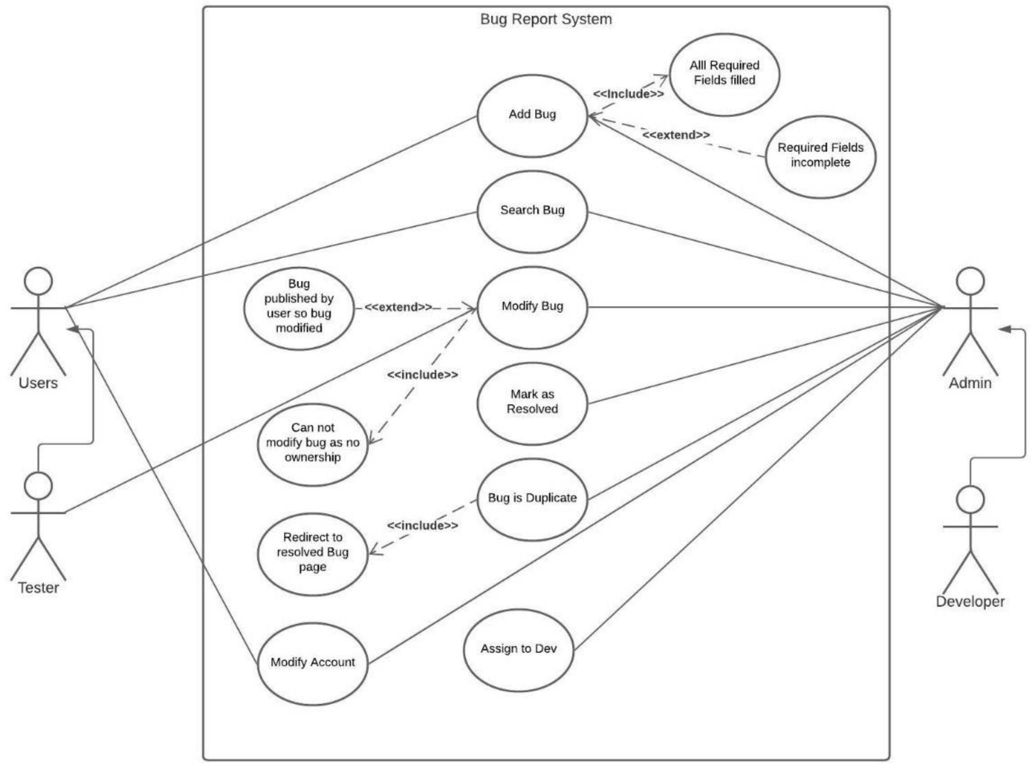
The following table identifies and describes the different users of BugVITa software. The information gathered about the different users of the system helped define what the software needs to do.Also, these users may be referenced in the requirements and diagrams.

Table of User Characteristics

User	**Description**
Tester	The tester is a user whose job is solely to use the software made by the developers and find bugs in the application. The tester will report the bug by simply adding a bug to the bug list by filling up it's name, description and other such attributes. The tester can be thought of as the general user in our case who has access to only the basic functionalities of bug reporting.
Developer	The developer is a user who has made the software or one who has been assigned to locate and fix the generated bugs and issues. A developer can also report bugs. Primarily, his job is to work on the bugs and coordinate with his team to make his software better.
Administrator	The admin can be thought of as the Super-User, the admin creates the project and is the leader in the development process of the application. The admin has complete control of the system and all privileges. The admin creates the project and adds 'users' linked to the project. Login credentials are generated and a user can login to start working on the project.
Analyst	The analyst is a part of the development team whose job is to assign a particular bug a

	developer if needed.The Analyst also has to peer review fixed bugs and clear them. An analyst has the same privileges as a developer with the addition of the ability to add/modify the 'assignee' for a particular bug

2.5


Bug Report System
Add Bug
Alll Required Fields filled
<<Include>>
<<extend>>
Required Fields incomplete
Search Bug
Bug published by user so bug modified
<<extend>>
Modify Bug
<<include>>
Can not modify bug as no ownership
Mark as Resolved
Bug is Duplicate
<<include>>
Redirect to resolved Bug page
Modify Account
Assign to Dev
Users
Tester
Admin
Developer

2.6 Entity Relationships

Figure 1 shows the entity relationships for the Electronic Stamp project.

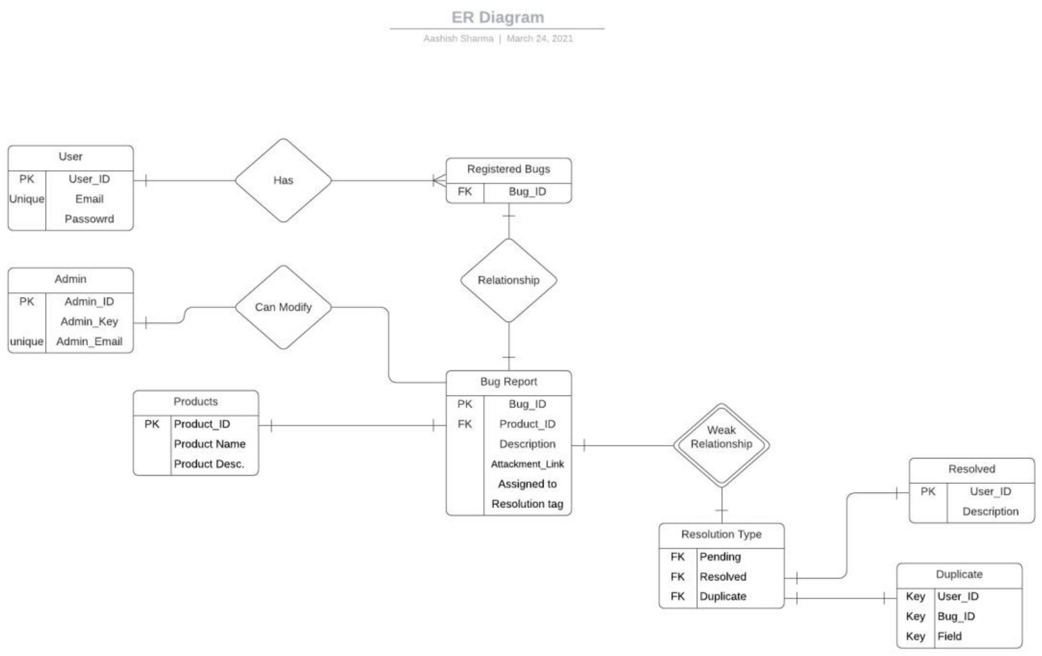


Figure 1 Entity Relationship Diagram

2.7 Data Flows

The following figures represent the data flow diagrams of BugVITa

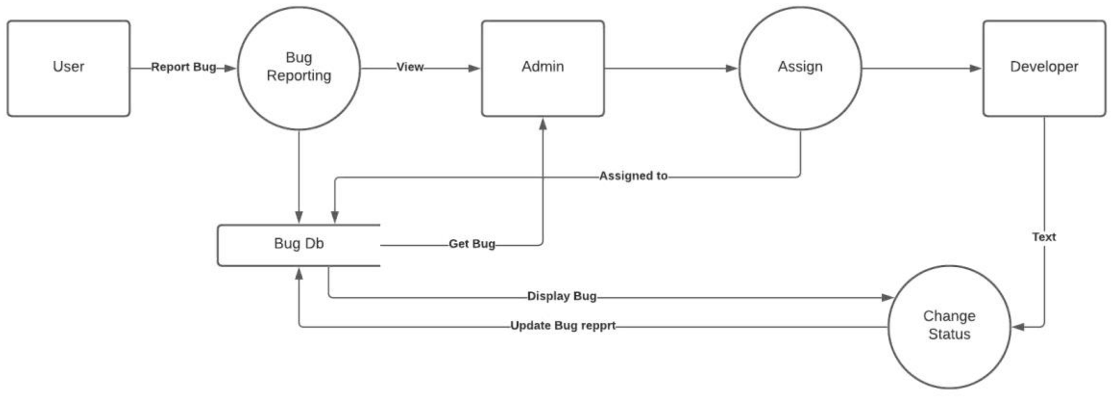


Figure 2 Data Flow Diagram

2.8 Data Dictionary

The follow tables in this section make up the data dictionary for the BugVITa project. Using the Data Flow Diagrams, the following Data Dictionary elements were defined.

- BugVITa
- Bug Report
- Search Page
- Users

Table of Data Dictionary

Data Dictionary Attribute	**Detail**
Name	BugVITa
Aliases	Bug management project
Where Used / How Used	Development phase Debugging code Bug Report by user Bug report by tester
Description	User = the primary stakeholder of the application has the privilege to report bugs and modify his bugs Tester = Person who deliberately looks for loop holes and logic faults in the code or a software responsible for robustness of the software Developer = Person who is responsible for either resolving or marking a bug report as duplicate when a bug is assigned

	Admin = super user which has privileges to look, manage and change database, also responsible for assigning a bug reported to and admin Product = A software that has been published or is in making that requires testing and/or bug resolution Bug report = an entity that has all the information about a bug and how it can be reproduced with the description of how the feature should perform

Data Dictionary Attribute	**Detail**
Name	Bug report
Aliases	None
Where Used / How Used	To store and display all the details of a reported bug by user
Description	Bug ID = a unique ID that is used to identify a bug report Bug Description = Any description related to the bug submitted by the user and how it is hindering their ability to use the product Product ID = the id of the product that is used to relate to the actual software the bug Attachments = additional images to help the developer

Data Dictionary Attribute	**Detail**
Name	User
Aliases	None
Where Used / How Used	Hierarchy of users
Description	Users = A person who uses the program Admin = a super user with privileges to modify and assign bugs to developer Developer = a user that is responsible for resolving th bug

Data Dictionary Attribute	**Detail**
Name	Search Page
Aliases	Bug Search
Where Used / How Used	To search a bug
Description	Search field = A text field to enter the query

	Results = Collections of results
Supplementary Information	More detail of email structure can be found in the specification of SMTP.

2.9 State Transitions

Please refer to Appendix A for the State Transition Diagrams. The first state transition diagram shows the state transitions when a Sender sends an email. The second diagram in Appendix A shows the state transitions when a Receiver receives an email or when a Sender receives a reply email. The last state transition diagram shows the state transitions when a Receiver sends a reply message.

Specific Requirements

2.10 System Features

2.10.1 Adding/Modifying a Bug

2.10.1.1 Introduction

BugVITa software shall allow a user to add a bug with a detailed description and some attributes in the Bug Editing page.
All users have the ability/privilege to add a bug

3.1.1.2 Bug Attributes:

- **Name**:- a name given to the bug / brief description of the bug.
- **Description**:- Provides complete description of the bug that occurs, output, expected output, input that caused the bug, etc.
- **Author:** Username of the user adding the bug (Automatically set)
- **Affects Version**:- Current version of software in which the bug is encountered.
- **Components**:- Other component affected due to bug. (Set only by Developer,Analyst,Admin)
- **Assignee**:- Person who is assigned to the bug.(Set only by Analyst/Admin)

  Assignee is a person who solves the bugs. So he is the actual resolver of the bug. A bug can be solved by an individual person or by a group of user.

- **Date**:- Day on which bug was last modified/created.(Set automatically)
- **Priority**:- Priority can be major, minor or critical (Set only by Developer,Analyst,Admin)
- **Issue Status**: (Can only be modified by Developer,Analyst or Admin)

  In model of bug tracking system there are different type of issue status like,

  - Open: (Default ,set at time of Bug Creation) . The bug is open for correction.
  - Reopened: The bug was closed previously but opened again.
  - Development in progress:- The bug solving is going on.
  - Waiting for peer review: The bug is solved and it is in queue to be reviewed by the tester.
  - Closed and Waiting for integration: The bug has passed peer review and needs to be integrated with the software and it is in waiting.

- Closed: The bug has been corrected and correction has been integrated.

2.10.1.2 Functional Requirements

Purpose: Creating/Modifying a bug report and saving it in the Database.

Input: Contents of bug report described as above.

Processing: Setting default values for attributes. Saving contents into the database.

Output: The bug report sent to the database.

2.10.1.3 Stimulus Response

User Actions	**System Actions**
(1) Compose/Edit Bug report	Open Bug Editing page.
	Set which fields are editable.
(2) Save bug report.	
	Close the Bug Report page
	Send bug report contents to database to be updated.
	Retrieve the entered Bug report by the user
	Display Bug Page for the bug entered by user.

2.10.2 Searching a Bug

2.10.2.1 Introduction

All users have the ability to search for a bug by providing keywords that search the
 'Name' , 'Description' , 'Assignee' , 'Author' and other Bug Attributes.
Matched results are returned to search page

2.10.2.2 Functional Requirements

Purpose: Searching for a particular Bug Report

Input: Text which contains the keywords separated by blank space for the Search process

Processing: Splitting text in keywords. Searching Database Product Table for matches. Computing no. of matches for each entry in the Table. Returning a list of results in descending order of number of matches.

Output: List of Most relevant bugs according to entered search parameters in decreasing order of relevance.(Most relevant on top)

2.10.2.3 Stimulus Response

User Actions	System Actions
(1)Enter Search parameters as Text	Receive Text
	Split text into list of keywords
	Search for keyword matches in each entry of Database Product Table
	Compute total number of matches for each entry
	Return list of search results sorted in descending order of matches
	Display results on Search 'Scene' with max of 10 to 15 bugs in one page. i.e,page 1,2,3 etc. as needed.
(2)Click on a bug	Go to that bug's Bug Page.

2.10.3 Viewing Overall Bug list

2.10.3.1 Introduction

This feature allows the user to view the list of all bugs in the Database Product Table.

2.10.3.2 Functional Requirements

Purpose: View all Bugs for a particular selected product.

Input: None

Processing: Retrieve Bug list from Database product table

Output: Show list of bugs

2.10.3.3 Stimulus Response

User Actions	System Actions
	(1) Retrieve Bug 'Name' 'Priority' 'Author' 'Date' 'Issue Status' for each bug entry and put in a list
	(2) Display list

2.10.4 View particular Bug Page

2.10.4.1 Introduction

Whenever we click on a bug, either from Search page or from Overall Bug List page,
It will go to the ‘Bug Page’ which shows all the details and attributes of the Bug report.

2.10.4.2 Functional Requirements

Purpose: View a particular Bug report.

Input: None

Processing: Retrieve that particular entry from the Database Product Table.

Output: All details and attributes of the Bug as described in 3.1.1.2

2.10.4.3 Stimulus Response

User Actions	System Actions
	Display all Bug attributes

2.10.5 Login/Signup

2.10.5.1 Introduction

User logs in using username and password to start working on his/her products.
User can also Sign up by entering desired credentials. After user signup, they will have to contact Administrator and ask admin to link their credentials to a desired product.

2.10.5.2 Functional Requirements

Purpose: Verifying if user exists in Database and logging them in if exists.

Input: Username and password.

Processing: Check Database User Table and verify user credentials and return User attributes. If user does not exist, return invalid

Output: Display Products page which shows products associated with user. If user does not exist, display popup telling user to sign up.

2.10.5.3 Stimulus Response

User Actions	System Actions
(1) Enter login Credentials - username and password.	Check if user exists in Database User Table
	Return that User does not exist if entry not found and stay at login page
	Else go to Products page and display all products associated with user
(2)Enter Signup data	(1)Check if both entered usernames and data match
	If doesn't match, clear fields and display 'Not matching'
	(2)If Matching, Check Database User Table if username already exists.
	If user already exists, display 'User already exists'
	(3)else if user doesn't exist,add credentials to Database and go to products page

2.11 Performance Requirements

The Bug Tracking System is a very essential tool when working on a large project with a lot of members in a team.
The following tables list the performance requirements that BugVITa should follow.

Table of Performance Requirements

Performance Requirement	**Description**
Database (Bugs, Users, Status)	The fast and secure database so that the bug report should be updated easily and faster.
Secure Database (Admin Data)	Protection at Admin level, should try and have security for unauthorized access of data.
Login and Sign Up API (Verifying Credentials)	Verification of credentials should happen very quickly without any compromise in security
Bug Report Submission API	The Bug Report Submission API must transmit data quickly and safely. Any failure in the API should not result in loss of data (Bug Report), rather the data must be retained and sent again.

2.12 Design Constraints

The Electronic Stamp project is extending the Pooka email software to implement the required functionality, which has placed certain design constraints on the design of the Electronic Stamp software. The table below lists those design constraints.

Table of Design Constraints

Design Constraint	Description

Bugzilla Design Paradigm	The features and design implemented by Bugzilla shall also be implemented and extended by our software BugVITa
Login/Signup Functionality	Bugzilla has a login/signup functionality to maintain security and provide authenticity to all bug reports. BugVITa will also being using the same paradigm to ensure a similar level of reliability.
Javascript/PHP/Java	Bugzilla being a web based site, is written in javascript/PHP/Java. Our software BugVITa will also use Javascript mainly to implement the various features.
Component Based/ Dynamic	A bug report system such as Bugzilla must maintain a dynamic sync with the backend to make sure bug reports are in sync with the database at all times.

2.13 Standards Compliance

The following table lists the different standards that the Electronic Stamp project is to be in compliance with.

Table of Standards Compliance

Standard	**Description**
Database	Our aim is to try and support the full SQL standard, but without making concessions to speed and quality of the code.

2.14 Software System Attributes

2.14.1 Reliability

Reliability in our software BugVITa will be ensured by thorough secure APIs, a safe database, and repeated testing. Necessary tests will be run on our software before finishing to ensure that BugVITa is completely reliable for all users. The source code will be thoroughly tested using the established test scenarios until the acceptance criteria are satisfied.

2.14.2 Security

The Electron Stamp software will utilize token encryption. Every login will generate a token from the backend, verifying the session. The backend will decrypt the password and verify the credentials with the database. On successful verification, a token will be generated, verifying the session on the frontend. This will ensure that our software BugVITa is completely secure.

2.14.3 Maintainability

The Electronic Stamp software extends the functionality of the Pooka email client software. The Pooka email client software is written in Java, a portable programming language. Because Pooka is written in Java, the Electronic Stamp software will also be written in Java. Java promotes good design practices due to the inherent structure of a Java program.

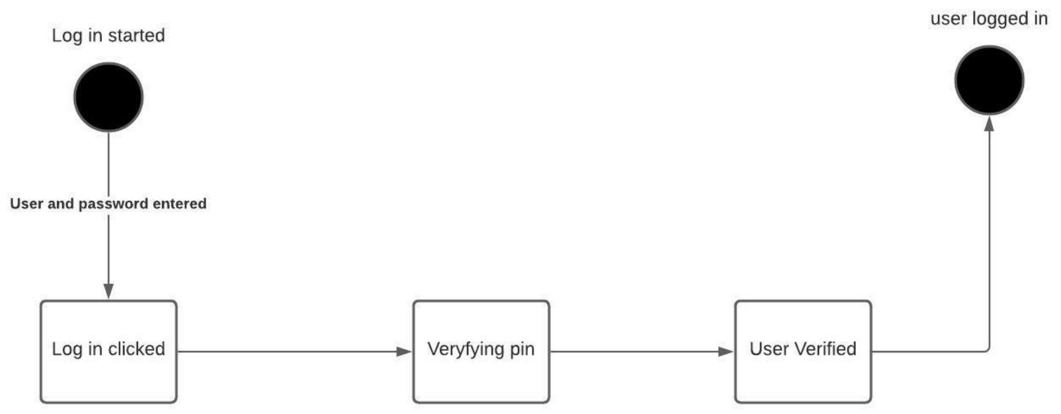
Along with the well-formed programming enforced by Java, best practice development conventions will be enforced for the construction of the Electronic Stamp software. These will include adequate commenting within the source code that complies with and uses the automated Java documentation standard so that source code documentation will be able to be automatically generated. Consistent variable naming conventions will be used by all the programmers. Consistent spacing will be used in the source code by all the programmers. The design of the source code will use the principles of Object-Oriented Design and the source code will be programmed using Object-Oriented Programming. Object-Oriented Design and Object-Oriented programming will make the code easier to understand.

2.14.4 Portability

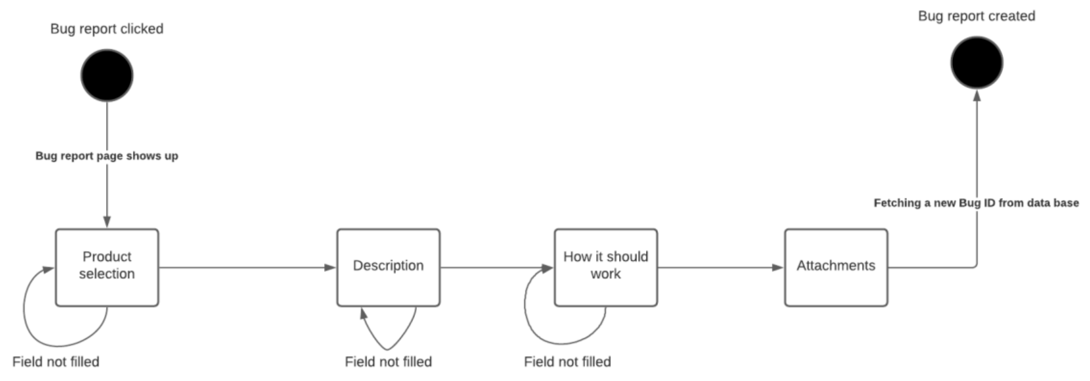
Our software BugVITa being a web based application/software provides a lot of portability since websites can be opened on any device from a browser. Our software will be made compliant and supporting of all browsers to ensure maximum portability.

It is safe to say that almost any device will be able to use our software without having to make any changes or installing any special platform/software.

Appendix A – State Transition Diagrams



State diagram for logging in



State diagram to create a bug page