
Helpdesk Design

Prepared by
NetiSoft Consultants Private Limited

Proposal Contact: Suresh Kumar Neti

Email: Suresh@NetiSoft.in

Phone: +91 98480 54551

All rights reserved. No part of this document may be used for any purpose, reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without written permission from NetiSoft Consultants Private Limited.

Contents

1. Introduction	3
1.1 Executive Summary	3
1.2 Abbreviations / Terms used	3
2. Features.....	4
2.1 User Roles.....	4
3. Database Design.....	5
3.1 State	6
4. Screens	7
4.1 Login	7
4.1.1 Screen.....	7
4.1.2 APIs called.....	8

1. Introduction

1.1 Executive Summary

The Helpdesk App is a mobile application designed for both iOS and Android to provide efficient and effective support to end-users in an organization. The app uses Python, Flask, React Native, and MySQL as its tech stack to provide a seamless experience for users. The app features a ticketing system that allows end-users to submit their support Tickets, and Service Providers to track and manage these requests and resolve them.

The Helpdesk App has three types of users: End Users, Service Providers and Organization Admins. End Users can submit tickets through the app, view their ticket status, and communicate with Service Providers assigned to their ticket. Service Providers can view and manage tickets assigned to them, update ticket status, and communicate with End Users. Organization Admins can manage Service Providers and assign the tickets to the service providers. They also have the complete access to the Ticketing System of the Organization and can perform all the functionalities of the End Users and the Service Providers.

The Helpdesk App is designed to improve the overall efficiency of support operations by streamlining the ticketing process and providing a platform for effective communication between End Users and Service Providers. It provides a scalable and secure solution that can adapt to the needs of any organization, regardless of size or complexity. The Helpdesk App's user-friendly interface and powerful features make it an essential tool for organizations looking to enhance their support operations.

1.2 Abbreviations / Terms used

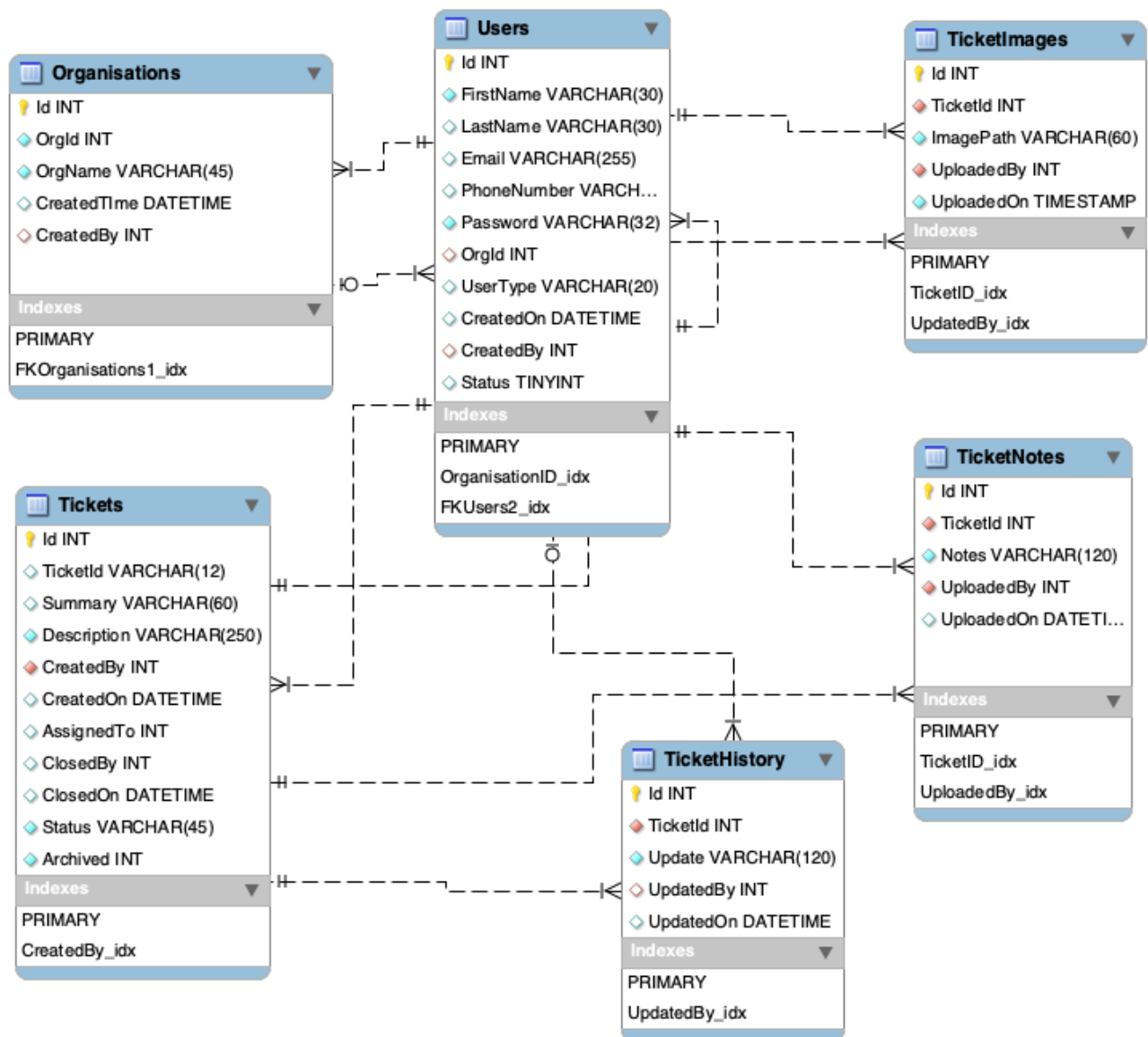
Term	Description

2. Features

2.1 User Roles

#	Role	Description
1	User	<ul style="list-style-type: none">• Primary Consumers of the App• Create, Update and View their Tickets• Track the Status of their Tickets• Close and Provide Feedback on the Tickets serviced
2	Service Provider	<ul style="list-style-type: none">• Update and View the List of Tickets Assigned to the Service Provider• Change the Status of the Tickets Assigned by Adding Comments or Notes and Uploading Pictures• Able to Access the Complete History of the Tickets Assigned to the Service Provider
3	Organization Admin	<ul style="list-style-type: none">• The Admin manually assigns the Tickets to the Service Provider in his Organization.• The Organization Admin should be able to create Service Providers, Users, and other Admins.• The Admin has all the Functionalities of the Users and the Service Providers for all the Tickets in the Organization.• The Admin can reassign the Ticket to another Service Provider.• Admin can access the List of all the Tickets for his Organization.

3. Database Design




3.1 State

#	Name	Description	Datatype	Remarks
1	OrgName	It is the Name of the Organization to which the User, Service-Provider or Admin belongs	VARCHAR(45)	Every User or Service Provider must have an Organization Name
2	FirstName	It is the First Name of the User	VARCHAR(30)	Mandatory
3	LastName	It is the Last Name of the User	VARCHAR(30)	Mandatory
4	Email	It is the Email of the user	VARCHAR(255)	UNIQUE
5	PhoneNumber	It is the Phone Number of the User	VARCHAR(11)	UNIQUE
6	Password	It is the Password of User's Account	VARCHAR(32)	Alphanumeric
7	UserType	It is type of the User which is currently using the app	VARCHAR(20)	The User can be Customer, Service Provider or Admin
8	Summary	It is the Summary of the issue raised by the User	VARCHAR(60)	
9	Description	It is the description of the issue raised by the User	VARCHAR(250)	Mandatory
10	AssignedTo	It has the ID of the Service Provider to whom the ticket has been assigned by the Admin	INT	It can be changed as Admin can assign Ticket to some other Service Provider
11	ClosedBy	It has the ID of the User who has Closed the Ticket	INT	It can be different from the ID of the User who Created the Ticket because it is possible that the Admin closes the Ticket.
12	Status	It shows the Current Status of the Ticket raised/issued by the User	VARCHAR(45)	The status can be Open, Assigned, WIP, Closed and Resolved
13	Update	It has the Updates regarding a Ticket.	VARCHAR(120)	This includes the Notes that are updated by the User and the Service Provider for a Particular Ticket.

4. Screens

4.1 [Login](#)

4.1.1 Screen



#	Field / Control	Validations	Remarks
1	User Id	Mandatory	Text Field
2	Password	Mandatory	Text Field
3	Organization	Mandatory	Text Field
4	Login		Button when Clicked Calls the API LoginUser
5	Forgot Password		Link, Opens the Forgot Password Screen
6	Register		Link, Opens the Register Screen

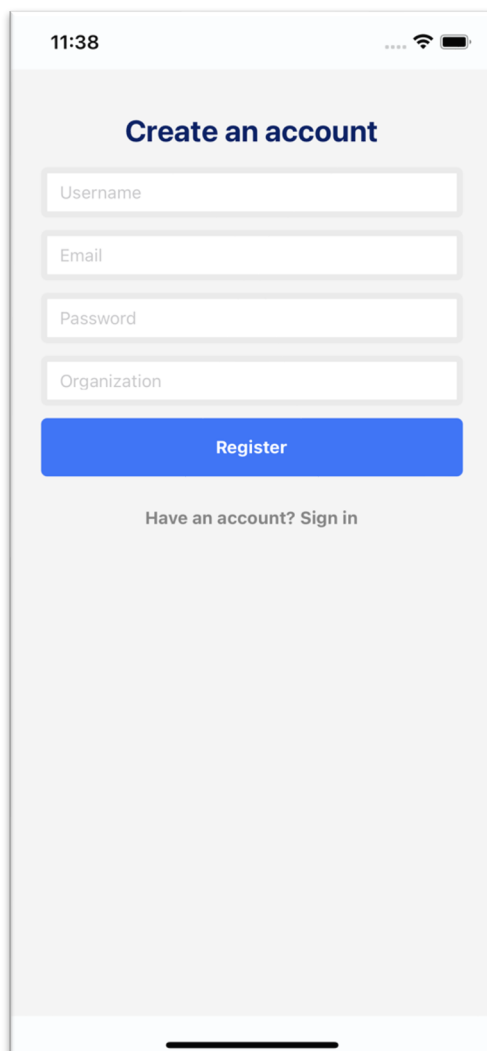
4.1.2 APIs called

- a) **LoginUser:** Checks whether the username, password, Organization combination entered is valid.

Input	{ "Username": "username", "Password": "password", "Organization": "organization" }
Output	User Id and JWT Token if Valid User, else returns 404 Error

4.2 [Register](#)

4.2.1 Screen



#	Field / Control	Validations	Remarks
1	Username	Mandatory	Text Field
2	Email	Mandatory	Text Field
3	Password	Mandatory	Text Field
4	Organization	Mandatory	Text Field
5	Sign In		Link, Open the Login Page
6	Register		Button, Calls the API Call RegisterUser

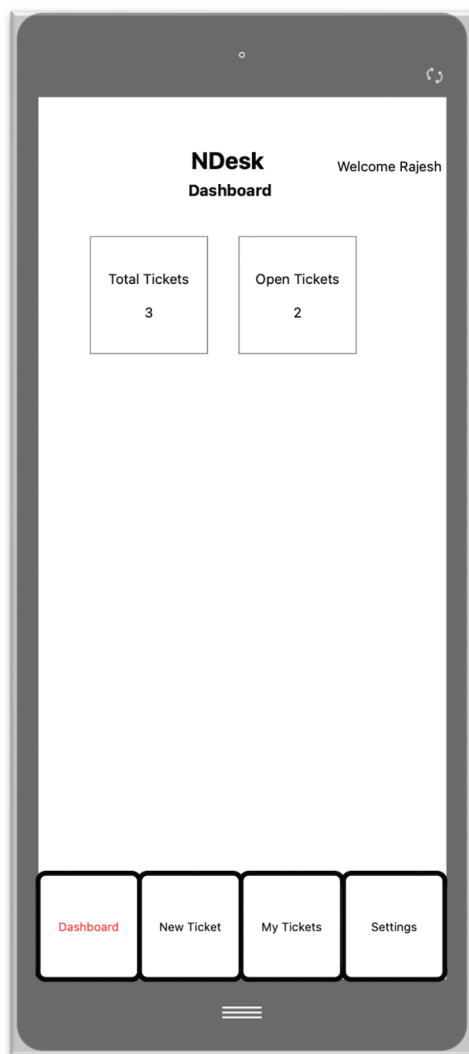
4.2.2 APIs called

1. **RegisterUser:** Register a New User in the Database.

Input	{ "Username": "username", "Email": "email", "Password": "password", "Organization": "organization" }
Output	User Id if Successfully Registered, else returns 404 Error

4.3 User Dashboard

4.3.1 Screen



#	Field / Control	Validations	Remarks
1	Total Tickets	Mandatory	Shows Total Number of Tickets Generated by the User using the API Call TotalUserTickets
2	Open Tickets	Mandatory	Shows Total Number of Tickets Generated by the User which are Currently Open using the API Call UserOpenTickets
3	Dashboard		Link, Opens the Dashboard Page
4	New Ticket		Link, Opens the New Ticket Page
5	My Tickets		Link, Opens the My Tickets Page

6	Settings	Link, Opens the Settings Page
---	----------	-------------------------------

4.3.2 APIs called

1. **TotalUserTickets:** Returns the Total Number of Tickets the User has Raised in entire lifetime.

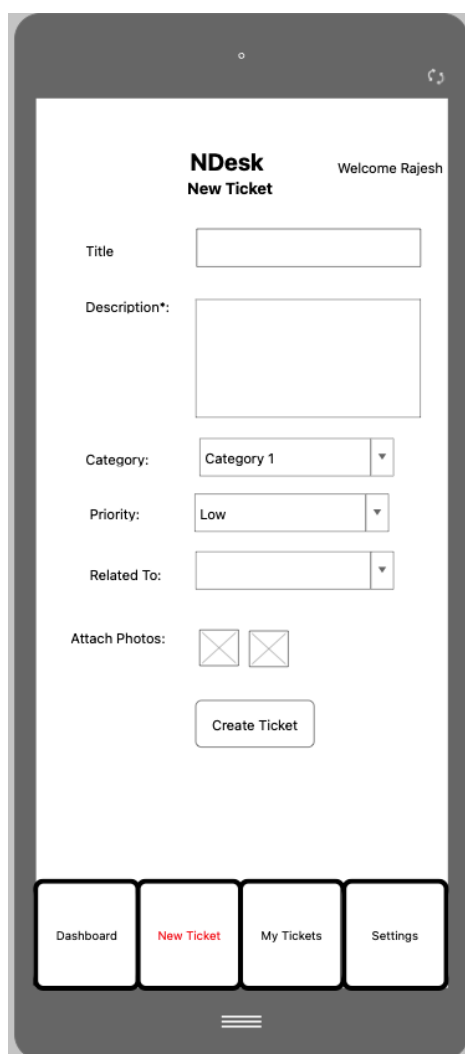
Input	{ "UserID": "id", "jwtToken": "token", }
Output	Total Tickets if Valid User, else returns 404 Error. { "TotalTickets": "totalTickets", }

2. **UserOpenTickets:** Return the Total Number of Open Tickets the User has Currently.

Input	{ "UserID": "id", "jwtToken": "token", }
Output	Total Open Tickets if Valid User, else returns 404 Error { "TotalOpenTickets": "totalOpenTickets", }

4.4 [New Ticket](#)

4.4.1 Screen



NDesk Welcome Rajesh
New Ticket

Title

Description*:

Category:

Priority:

Related To:

Attach Photos:

Dashboard **New Ticket** My Tickets Settings

#	Field / Control	Validations	Remarks
1	Title	Mandatory	Text Field
2	Description	Mandatory	Text Field
3	Category	Mandatory	Drop Down (Choose One Option out of Multiple Choices)
4	Priority	Mandatory	Drop Down (Choose One Option out of Multiple Choices)
5	Related To	Mandatory	Drop Down (Choose One Option out of Multiple Choices)

6	Attach Photos		Button, when Clicked calls UploadPicture API
7	Create Ticket		Button, when Clicked calls CreateTicket API
8	Dashboard		Link, Opens the Dashboard Page
9	New Ticket		Link, Opens the New Ticket Page
10	My Tickets		Link, Opens the My Tickets Page
11	Settings		Link, Opens the Settings Page

4.4.2 APIs called

1. **CreateTicket:** Raises a Ticket for the Particular User with the Open Status.

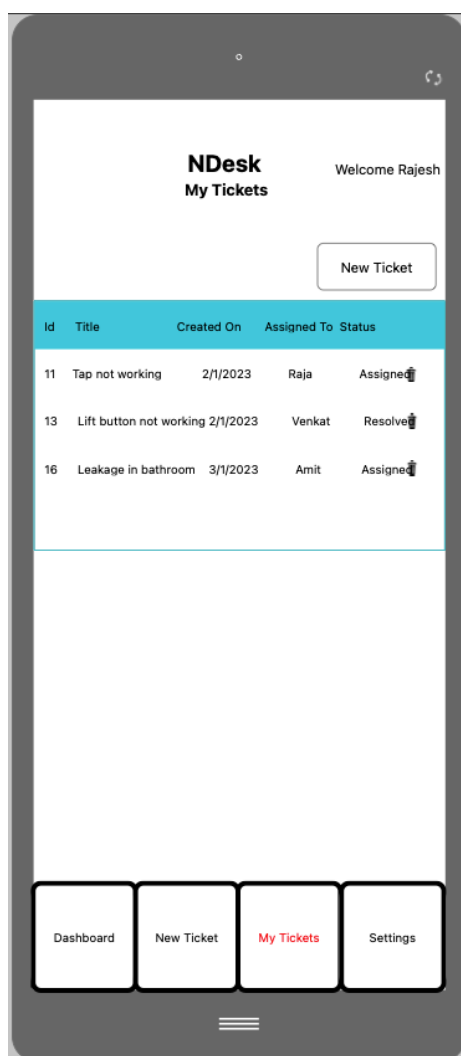
Input	{ "UserID": "id", "jwtToken": "token", "Title": "title", "Description": "description", "Category": "category", "Priority": "priority", "RelatedTo": "relatedTo" }
Output	Status Code 200 if Valid User and Valid Ticket Details, else returns 404 Error

2. **UploadPicture:** Uploads a Picture to the Corresponding Ticket.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID": "ticketID" }
Output	Status Code 200 if Valid User and Valid Ticket Details, else returns 404 Error

4.5 My Tickets (User)

4.5.1 Screen



#	Field / Control	Validations	Remarks
1	Title	Mandatory	Shows Title for Each of the Tickets using API Call TicketDetails
2	Created On	Mandatory	Shows Date of Creation for Each of the Tickets using API Call TicketDetails
3	Assigned To	Mandatory	Shows Service Provider Assigned for Each of the Tickets using API Call TicketDetails
4	Status	Mandatory	Shows Current Status for Each of the Tickets using API Call TicketDetails

5	Id	Mandatory	Shows ID for Each of the Tickets using API Call TicketDetails
6	Id		Link, Open the View Ticket Page for that Particular Ticket ID
7	New Ticket		Link, Opens the New Ticket Page
8	Dashboard		Link, Opens the Dashboard Page
9	New Ticket		Link, Opens the New Ticket Page
10	My Tickets		Link, Opens the My Tickets Page
11	Settings		Link, Opens the Settings Page

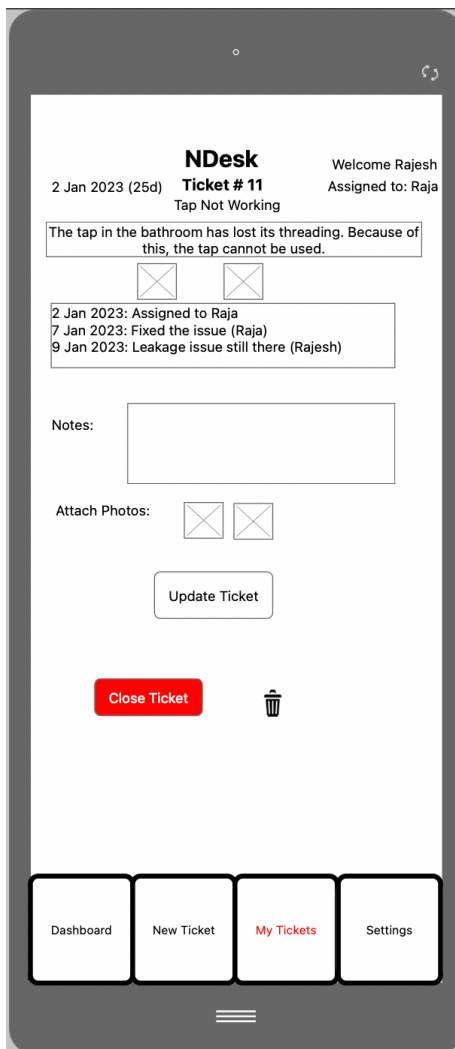
4.5.2 APIs called

1. **TicketDetails:** Returns the Details of the All the Tickets for the Particular User.

Input	{ "UserID": "id", "jwtToken": "token" }
Output	Return the Details for all the Tickets if Valid User, else returns 404 Error { "Title": "title", "CreatedOn": DATE, "AssignedTo": "assignedTo", "Status": "status", "TicketID": "tickedID", }

4.6 Ticket Details (User)

4.6.1 Screen



NDesk Welcome Rajesh
2 Jan 2023 (25d) **Ticket # 11** Assigned to: Raja
Tap Not Working

The tap in the bathroom has lost its threading. Because of this, the tap cannot be used.

2 Jan 2023: Assigned to Raja
7 Jan 2023: Fixed the issue (Raja)
9 Jan 2023: Leakage issue still there (Rajesh)

Notes:

Attach Photos:

Update Ticket

Close Ticket

Dashboard New Ticket My Tickets Settings

#	Field / Control	Validations	Remarks
1	Date	Mandatory	Shows the Ticket Creation Date using the API Call SpecificTicketDetails
2	Ticket ID	Mandatory	Shows the ID of the Ticket using the API Call SpecificTicketDetails
3	Assigned To	Mandatory	Shows the Name of the Service Provider to whom the Ticket was Assigned using the API Call SpecificTicketDetails
4	Summary	Mandatory	Shows the Summary of the Ticket

5	Description	Mandatory	Text Field, can be Updated
6	History	Mandatory	Shows the History of the Ticket. This includes the Notes along with the Time Stamps which are updated by the Service Provider or the User in the Ticket Details Section.
7	Notes		Text Field, Updates the History of the Ticket
8	Attach Photos		Button, Uploads Picture from Local Device using the API Call UpdatePictures
9	Update Ticket		Updates the details of Ticket using the API Call UpdateUserTicket
10	Delete Ticket		Button, Deletes the Ticket using the API Call DeleteTicket
11	Close Ticket		Button, Changes the State of the Ticket to Closed using the API Call CloseTicket
12	Dashboard		Link, Opens the Dashboard Page
13	New Ticket		Link, Opens the New Ticket Page
14	My Tickets		Link, Opens the My Tickets Page
15	Settings		Link, Opens the Settings Page

4.6.2 APIs called

1. **UpdateUserTicket:** Updates a Ticket for the Particular User with the Particular Ticket ID.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID": "tickedID", "Title": "title", "Description": "description", "History": "history", "Notes": "notes", }
Output	(Status Code 200) Return the Details for all the Tickets if Valid User and Valid Ticket Details to be Updated, else returns 404 Error

2. **DeleteTicket:** Deletes a Ticket for the Particular User with the Particular Ticket ID.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID": "tickedID", }
Output	(Status Code 200) Deletes the Ticket Details from the Database if Valid User and Valid TicketID, else returns 404 Error

3. **CloseTicket:** Changes the Status Code of the Particular Ticket to Closed.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID": "tickedID", }
Output	(Status Code 200) Changes the Status Code of the Ticket to Closed if Valid User and Valid TicketID, else returns 404 Error

- 4) **UploadPicture:** Uploads a Picture to the Corresponding Ticket.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID": "ticketID" }
Output	Status Code 200 if Valid User and Valid Ticket Details, else returns 404 Error

- 5) **SpecificTicketDetails:** Returns the Details of the Particular Ticket for the Particular User.

Input	{ "UserID": "id", "jwtToken": "token", "TicketID" : "ticketID", }
Output	Return the Details for the Ticket if Valid User, else returns 404 Error { "CreatedOn": DATE, "AssignedTo": "assignedTo", "Status": "status", "TicketID": "tickedID", }