**IT Help Desk Ticketing System**

**Final Report**

4 May 2022

Team: Dalton Coughlin

Grant Collins

Karina D'Abbraccio

Jared Rice

James Sweatt

Kobi-yoshi Hsu

**Contents**

1. Overwiev...................NOT DONE
2. .....................................3

2. Goals............................................................................................................................3

3. Product Requirements..................................................................................................3

4. Out of Scope………………………………………………………………………………….6

5. Open Questions……………………...…...………………………………………………….6

6. Security and Privacy………………………………………………………………………....6

7. System Specification….................................................................................................6

8. Timeline and Milestones…………….............................................................................7

9. Responsibilities.............................................................................................................7

1. **Overview**

Support automation platforms behave as a management system to reduce response time, enhance service management, assist with its support and its ticketing, provide ticket routing as well as help with prioritization of customer requests.  Many companies suggest similar software with diverse functionality. Because of the tight schedule, the class project includes the basic functionality: end-user interface, may submit and check tickets; tech interface, may assign to himself and check tickets; message possibilities between tech and end user.

The user version may be found at [**https://ithelpticket.net/**](https://ithelpticket.net/), usernames are **'student-user***'* and **'student-tech'** both with a *password* of '**umgcCMSC495**'.

*Updates on the documents* include:

* Project Plan: added table of contents, expanded Out-Of-Scope section;
* Design: added table of contents, current User Interface design in the section High Level and User Interface Design;
* Product Specification: added table of contents;
* Test Plan and User’s Guide: expanded User’s guide with all current steps separately for Tech and End user, supported by screen captures with the current design.

The *problems* of a team through the development process were lack of knowledge and experience in working on a group project, skills gaps in all system’s elements of the product, communication and scheduling of meetings and updates.

Even with the problems listed, all the functionality listed in Requirement Specifications was *developed successfully*. The product satisfies all set criteria. As it was noted in the Open Questions section of the Product Specifications Document, the initial UI changed; the ‘Create User’ option was not added, password and user are created by the system administrator.

*Summary of individual contributions:*

Dalton Coughlin:

* Managed team
* Database design
* Project design
* Assisted with coding
* Mentored and assisted team members

Kobi H.:

* Setup Github
* Sorted tickets by priority on home page
* Date formatting on home page

Karina D’Abbraccio:

Jared’s Rice:

* Database Management
  + Designed database scheme
  + Create models in Django to represent database
  + Utilize Django’s user model and implement it into our project
  + Create usable data for development and testing
  + Assist team members with creating local PostgreSQL server for development
* Host Server
  + Set up AWS ECS server to host web app
  + Set up AWS RDS server to host PostgreSQL database
  + Set up domain name with SSL certificates
  + Configured and deployed app with WSGI and NGINX
* Backend Development
  + Created queries for gathering individual ticket information
  + Created queries for displaying comments associated with tickets
  + Auto-assigns ‘Users’ to ticket when creating
  + Allowed ‘Techs’ to create a ticket on ‘User’s’ behalf
  + Allowed ‘Techs’ to assign/un-assign themselves from a ticket
  + Allowed ‘Techs’ to close/re-open tickets
* Frontend Development
  + Created Add Comment page
  + Created Assign/Un-Assign Ticket page
  + Created Close/Re-Open Ticket page
  + Hides ‘Tech’ only functions from “Users’

James Sweatt:

* Worked on bug fixing
* Added ticket querying functionality on home page with all tickets and my tickets
* Adjusted and built javascript functionality for client to server querying
* Adjusted html pages to include home links and display of information for tickets
* Adjusted home page to separate tech and user tickets on login
* Worked with other students on bug fixing and changes for project

Grant Collins:

* Assisted in the creation of the project plan and user interface design
* Researched the Django framework and how to use it in the frontend development process
* Created rough drafts of HTML templates along with the Cascading Style Sheets
* Assisted with finalizing the template and CSS designs
* Assisted with the programming for frontend functionality and user interface
* Assisted with the data formatting in the display
* Tested the program’s functionality

1. **Project Plan** 
   1. **Overview**

Support automation platforms behave as a management system to reduce response time, enhance service management, assist with its support and its ticketing, provide ticket routing as well as help with prioritization of customer requests. To deliver superior customer satisfaction, ensuring you invest in the best helpdesk ticketing system available is an absolute must for any business that wants to deliver perfect customer experience.

The project would be comprised of two users, an end user and a help desk support tech. An end user would be able to submit tickets, keep track of tickets, and send messages to the techs who have their ticket. A help desk support tech would be able to assign themselves a ticket, send messages to the end user, and change the status of the ticket.

* 1. **Objectives**

The application will include:

1) end-user interface, may submit and check tickets;

2) tech interface, may assign to himself and check tickets;

3) message possibilities between tech and end user;

* 1. **Project Scope**

A helpdesk ticketing system is software utilized by a customer support team to develop, manage, and maintain lists of customer issues without the need for a call center. End-users will be able to experience a high-quality response to customer requests – even through self-service options. Certain limitations of this project define the functionality of a final product, that should include the end-user interface, tech-user interface, and possibility of their communication. The tickets should be stored and available for both users as defined in specifications.

*The completed project should allow users to:*

* Login for all users
* End user: open tickets, close tickets
* Tech User: assign tickets, close tickets, open tickets
* Tickets: be stored, updated, sorted
* Chat between end-user and tech-user.

The project is completed with the purpose of education and team collaboration experiences. The scheduled workflow and possibility to have everyone involved and participating is also a sign of success for this project.

* 1. **Out of scope**

The system at the end of the project will not include following functions:

- reports or other metrics on tickets;

- create pre-written responses;

- use automatic ticket assignment for even distribution;

- accept messages from different channels;

- provide tutorials or answers on frequently asked questions;

- create and send customer satisfaction surveys;

- customized dashboards;

- other functions not mentioned in the Project Scope section.

There are no plans to integrate this application is other programs. No additional privacy protection or security is provided.

* 1. **Deliverables**

For the project, deliverables include: requirements specifications, design documentation, source code, test results, user’s guide.

* 1. **Assumptions and Constrains**

Project will be written in Python programming language.

Schedule defined by the Key Milestones section and Budget may not be changed. There are no additional human resources available, but the responsibilities may be changed if needed.

* 1. **Resource Management**

**Team** for the project is: Dalton Coughlin, Grant Collins, Karina D'Abbraccio, Jared Rice, James Sweatt, Kobi-yoshi Hsu.The table below represents initial responsibilities for the project.

|  |  |
| --- | --- |
| **Person** | **Responsibilities** |
| Dalton Coughlin | Project Lead |
| Grant Collins | Frontend Developer |
| Jared Rice | Database Admin |
| James Sweatt | Backend Developer |
| Kobi-yoshi Hsu | Tech Lead, Server Management |
| Karina D’Abbraccio | UI/UX Designer, Tech Documents |

**Budget**: any payments are not expected to be received for the participation in the project.

**Time**: the final project with all documentation to be delivered by May, 8, 2022 and to be submitted by all team members by May,10, 2022.

Each team member may use the result of the group work, with credit to the other participants.

* 1. **Key Milestones and Timelines**

|  |  |  |
| --- | --- | --- |
| Date\* | Responsible person(s)\*\* | Task |
| March 27 | Dalton C., Kobi-yoshi H.  Karina D., **all** members | Specifications,  Project Plan |
| April, 2 | Dalton C., Kobi-yoshi H.,  Karina D. | User’s Guide, Test plan |
| April, 8 | Dalton C.,  Kobi-yoshi H.  Karina D. | Design: classes, methods, fields, interfaces, data structures, IO formats.  Finalize test cases |
| April 15 | Dalton C., Kobi-yoshi H. | Software – at least 60% of functionality |
| April 22 | Dalton C., Kobi-yoshi H. | Software – all functionality |
| May 1 | Dalton C., Kobi-yoshi H.  Karina D. | Software – tested and fixed.  Changes to previous documents |

\* Timeline is based on the class schedule to allow each group member to become familiar with the result and leave comments to the group/responsible persons.

\*\* The person responsible for the output might still request smaller tasks from the other team members.

Each member of the team shall conduct a review of documents and software that is being submitted.

***Software Development Plan:***

Dev Team : Kobi (Lead), Grant, James, Jared, Dalton;

Responsibilities: execute user stories, update technical documents.

|  |  |  |
| --- | --- | --- |
| **Date** | **Task** | **Status** |
| March 29 | * use Project Requirements to setup code repository * build docker containers for deployment | done  done |
| March 31 | * setup user sessions * complete index page | done |
| April 7 | * complete login process * complete ticket submission process |  |
| April 14 | * setup CI/CD pipeline * complete ticket management process * complete ticket board |  |
| April 21 | * setup E2E tests * system test |  |
| May 1 | * completed software |  |

***Frontend*** (Grant Collins)

|  |  |
| --- | --- |
| **Date** | **Task** |
| March 29 | - Create project plan  - Research Django Framework |
| April 5 | - Rough draft of HTML template designs |
| April 10 | - Finalize template designs and send out for acceptance  - Start working on the ways to display the data from the DB on the page |
| April 15 | - Work on backend to frontend functionality |
| April 21 | - Cleanup front end designs and functionality |
| May 1 | - Fully tested project and make adjustments as needed |

***Database*** (Jared Rice)

|  |  |
| --- | --- |
| **Date** | **Task** |
| March 29 | - Create project plan  - Coordinate with Kobi for server details and system requirements  - Coordinate with Dalton and James on backend system requirements  - Choose an appropriate SQL database to meet our needs  - Research and get spun up on possible programs |
| April 3 | - Rough draft of database schema  - Develop test data for tables with group |
| April 7 | - Finalize database schema  - Coordinate with Kobi and install all required software, correct versions  - Create database and configure backend environment for connection |
| April 14 | - Initialize database tables and columns  - Coordinate with James for proper backend/database queries |
| April 21 | - Insert test data into tables for testing  - Coordinate with James for proper backend/database queries |
| May 1 | - Fully tested project and make adjustments as needed |

***Backend*** (James Sweatt)

|  |  |
| --- | --- |
| **Date** | **Task** |
| March 29 | -Research views, urls, and database queries |
| April 3 | -Setup urls and views for login process |
| April 7 | -Setup urls and views for ticket submission process |
| April 14 | -Setup urls and views for ticket management process for end user  and tech |
| April 21 | -Coordinate with frontend developer and database admin for testing |
| May 1 | - Fully tested project and make adjustments as needed |

***Design***

Login and registration page is to be completed by April 1, all page’s design is to be completed by April 8. (Dalton C., Karina D.).

Communication and meeting times to be arranged as needed.

* 1. **Risk Assessment**

*Project Scope* - Tiny adjustments here and there can add up to hours of additional work. Probability: low.

To prevent scope risk, the minimal objectives set and change policy is agreed and presented in the next section. Depending on the change suggested, it may be possible to accommodate a bit of scope creep, but such additions will result in changes to the timeline of the project.

*Resources* (personnel, financial, and physical) – Some team members may appear not to be able to finish their task. Probability: average.

The other team members may take the responsibility to do that task. If the previous option is not available, the project scope may be reduced in case it is possible to deliver functional software.

*Project delays* – school due dates don’t allow to submit work later, therefore the **last due day minus one day is the deadline** to deliver any available result to the group. Probability: low.

*Failures of Technology or Communication* – this might be the first group project for some students, no work hours are set for the online class, team includes people from the different time zones, some may have their job and other schedule, so communication and co-working is estimated to be the risk for this project. Probability: high.

Beyond the method of communication, expectations about response times and set a good example of professional communication style and tone should be setup by the beginning of development process.

* 1. **Change Management**

All changes to the functionality are to be discussed and agreed with each group member who is affected by such changes and agreed by project manager and tech lead.

1. **Requirements Specification**

3.1. End user should be able to:

- view all tickets;

- submit new ticket;

- contact tech responsible for the ticket /already or will be assigned later;

- filter tickets on status, assigned, priority;

3.2. Tech support user should be able to:

- filter tickets on status, assigned, priority;

- assign himself a ticket that has not been assigned to anyone;

- contact end user of his (techs’) tickets;

3.3. Database to store tickets.

Each ticket should have: id, date entered, date due,

Status, last checked,

assigned tech,

body (text description)

comments.

3.4. Login and logout for all users.

1. **System Specification**

The project is web-based application so any PC that is able to run any web browser such as Firefox or Google Chrome will be able to use the application. The view in different browsers or on mobile devices is not supposed to be tested, so there is a chance that design might vary not substantially, and this doesn't affect functionality.

Hardware:

Web Server: AWS EC2 virtual server on the t2.micro tier

Database Server: AWS RDS virtual db on the db.t3.mucro tier

Performance:

Web Server: 1 vCPU @ 2.5 GHz, 1GB memory, 8GB storage

Database Server: 2 vCPU @ 2.5 GHz, 1 GB memory, 20 GB storage

1. **User's Guide**

The application includes end-user and tech user interface, they may submit and check tickets, leave comments on specific tickets.

1. *Go to* <https://ithelpticket.net/> in your internet browser. You should have internet access.
2. ***Login***with the valid username and valid password, typing them in corresponding text fields of the login page, and press the button “Submit”. The user will be directed to home page of end user or tech user.

*Graphical user interface, application

Description automatically generated*

The link in the footer GitHub Repository will open the page on the GitHub where the source for the project is located.

1. ***This section is for the Tech user.*** Tech userwould see the home page with the available menu options on the left side: My Tickets, All Tickets, Create Ticket, Filter (Search, Status, Assigned, Priority), and preview for all tickets.

Graphical user interface, application

Description automatically generated

3.1. Mouse hover over the ticket will highlight the ticket in light blue color, mouse click on highlighted ticket will show the *details on this ticket*:

Graphical user interface, text, application

Description automatically generated

Along with details, there are comments or a message if the ticket does not have any comments, links to add comment, close ticket, assign to ticket.

3.2. To *Add comment*, user need to fill the text area and click the “Submit” button:

Graphical user interface, text, application, chat or text message

Description automatically generated

Then the comment will appear in the Ticket details:

*Graphical user interface, text, application

Description automatically generated*

3.3. If the ticket is not assigned to anyone, the tech may *assign himself* a ticket by clicking Assign to ticket in the Ticket Details:

Graphical user interface, text, application

Description automatically generated

The ticket’s status will be Assigned, and the link on the bottom will change to *Unassign from Ticket:*

Graphical user interface, application

Description automatically generated

3.4. To unassign from ticket, the tech user needs to click the link *Unassign From Ticket* and click Yes:

*Graphical user interface, text, application

Description automatically generated*

3.5 The Tech User may *close* ticket by clicking the link on the bottom of the ticket details and clicking “Yes” button:

A picture containing application

Description automatically generated

The status of the ticket will change to Closed.

3.6. Clicking *My Tickets*, the tech will see the tickets assigned to them, All Tickets will show all tickets (as on the Home page).

Home Page – shows all tickets sorted by priority in descending order.

All tickets – shows all tickets sorted by their number.

3.7. To *search* for ticket input the ticket number (as on screen capture above) or the key words (as on the screen capture below):

*Graphical user interface, application

Description automatically generated*

*Graphical user interface, text, application, Teams

Description automatically generated*

*3.8. Create ticket.* Tech user can create ticket for the user by clicking the “Create Ticket” from the menu. In the new window/page the text areas must be not empty and describe the short summary of the problem and provide description. The dropdown menu allows to set priority on routine / urgent / emergency, and choose the User for this ticket. After this, press the button “Submit”. Tech user sets the status, user to be assigned to.

Graphical user interface, application

Description automatically generated

3.9. *Filter tickets.* Press the menu option “All”, “Open”, “Closed” to see corresponding tickets.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text

Description automatically generated Graphical user interface, text, application

Description automatically generated

If in the other fields for sorting the criterion is selected instead of “All”, the tickets will be filtered to satisfy all selected criteria.

1. ***This section is for the End user.*** End userwould see the home page with the available menu options on the left side: My Tickets, Create Ticket, Filter (Search, Status, Assigned, Priority), and preview for all tickets. Graphical user interface, application

   Description automatically generated

3.1. Mouse hover over the ticket will highlight the ticket in light blue color, mouse click on highlighted ticket will show the *details on this ticket*:

Graphical user interface, application

Description automatically generated

Along with details, there are comments or a message if the ticket does not have any comments, links to add comment, close ticket, assign to ticket.

3.2. To *Add comment*, user need to fill the text area and click the “Submit” button:

Graphical user interface, text, application, chat or text message

Description automatically generated

Then the comment will appear in the Ticket details:

*Graphical user interface, text, application

Description automatically generated*

3.6. Clicking *My Tickets*, the end will see the tickets submitted by them sorted by their number. Home Page – shows all tickets sorted by priority in descending order.

3.7. To *search* for ticket input the ticket number (as on screen capture above) or the key words (as on the screen capture below):

Graphical user interface, text, application

Description automatically generated

*Graphical user interface, text, application

Description automatically generated*

(This User does not have ticket number 5).

*3.8. Create ticket.* Tech user can create ticket for the user by clicking the “Create Ticket” from the menu. In the new window/page the text areas must be not empty and describe the short summary of the problem and provide description. The dropdown menu allows to set priority on routine / urgent / emergency. After this, press the button “Submit”.

Graphical user interface, text, application

Description automatically generated

3.9. *Filter tickets.* Press the menu option “All”, “Open”, “Closed” to see corresponding tickets.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text

Description automatically generated Graphical user interface, text, application

Description automatically generated

If in the other fields for sorting the criterion is selected instead of “All”, the tickets will be filtered to satisfy all selected criteria.

4. At the end of working, click ***Logout*** in the page header

Graphical user interface, application

Description automatically generated with medium confidence

1. **Test Plan** 
   1. **Test Scope**

As required by the Product Specifications, the project only focus on testing all the **functions** and external interface of website**IT Help Desk Ticketing System**.

In-scope items: functional testing. Testing will be focused on:

* API created for the application;
* Integration test;
* System test.
  1. **Out of** **scope**

Nonfunctional testing such as**stress, performance**or**logical database** currently will not be tested.

Out-of-scope items: hardware, and other external interfaces.

* 1. **Risks and issues**

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Lack the required skills for testing | allocate additional time for additional research on testing methods |
| Lack of time for appropriate testing due to tight schedule | Set priority to each test activity |
| A lack of cooperation negatively affects your employees’ productivity | Encourageeach team member in their task, and inspire them to greater efforts |
| Financial risks | Doesn’t apply |

* 1. **Test Objectives**

Final goal is to deliver bug free product that is required by Product Specifications by the due date of a project.

* 1. **Software features that need to be tested**

Check whether functionality is working as expected without any errors or bugs. The testing should cover following: account (sign in, login, logout), add ticket, sort tickets, close ticket, update status on ticket, ticket attributes, add comment to the ticket.

Check that only authorized users may use the product. This includes testing the login of tech and end users and their access to the correct functionality as result.

Check that the external interface of the website such as **UI** is working as expected and meet the customer need. The testing should cover all web elements (button, menu, text), HTML/CSS.

Verify usability for convenience for the user.

* 1. **Testing Schedule**

|  |  |
| --- | --- |
| **Date** | **Task** |
| April 3 | - Develop test data for tables with group |
| April 7 | * check login and registration process * check that database is reflecting all needs of the project |
| April 14 | * check home page |
| April 17 | * check database tables and columns * check proper backend/database queries |
|  | * setup E2E tests * check front end designs and functionality * system test |
| April 21 | - Insert test data into tables for testing   * Check proper backend/database queries |
| **May 5** | Fully tested project |

* 1. **Resource Planning**

Unit/ Code Functional Testing will be done by the development team.

The final testing will include all scenarios from the user guide, from perspective of end user and tech user.

Test data will be loaded directly into the database: 15 users and 5 techs, with 50 tickets total and 100 comments. The data will be exported to an excel sheet and sorted, and the testing results will be validated.

* 1. **Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Number: | Test Case: | Test Parameters: | Expected Output: | Actual Output: | Pass: |
| 1.1 | Valid Login By Tech | Correct username/password | Able to Login and directed to home page | Tech logged in and directed to home page | Yes |
| 1.2 | Invalid Login By Tech | Missing or incorrect password | Not able to login, error given | Tech not logged in and error was given | Yes |
| 1.3 | Valid Login By User | Correct username/password | Able to Login and directed to home page | User logged in and directed to home page | Yes |
| 1.4 | Invalid Login By User | Missing or incorrect password | Not able to login, error given | User not logged in and error was given | Yes |
| 2.1 | Creation of Ticket by User | All fields entered with information | Ticket shows on home page, direct to home page | User able to submit ticket | Yes |
| 2.2 | Creation of Ticket by Tech | All fields entered with information | Ticket shows on home page, user assigned and is\_assigned is true | Tech able to submit ticket | Yes |
| 2.3 | Invalid/Missing inputs on ticket creation | Missing information on fields | Error shown | Tickets were created | No |
| 3.1 | All tickets show for Tech | Tech logged in | All tickets should show for tech | All tickets should show for tech | Yes |
| 3.2 | Only user tickets show for User | User logged in | Only user tickets should be shown | Only user tickets should be shown | Yes |
| 4.1 | Status filter | Status filter set to one of two statuses | Filtering status is filter | Filtering status is filter | Yes |
| 4.2 | Assigned filter | Assigned filter set to one of two statuses | Assigned status is filtered | Assigned status is filtered | Yes |
| 4.3 | Priority filter | Assigned filter set to one of three statuses | Priority status is filtered | Priority status is filtered | Yes |
| 4.4 | My Tickets functioning | Button clicked for My tickets | Shows only current user's tickets | Shows only current user's tickets | Yes |
| 4.5 | All tickets functioning | Button clicked for All tickets | Shows all tickets | Shows all tickets | Yes |
| 4.6 | Search functioning | Search input filled out | Search shows the ticket information by ticket number or title | Search shows the ticket information by ticket number or title | Yes |
| 5.1 | Ticket information showing | Ticket is cliked on home page | Ticket information should show correct information on click of ticket | Ticket information is shown | Yes |
| 6.1 | Add comment | Comment information filled out | Comment should be added to ticket | Comment was added to ticket | Yes |
| 6.2 | Invalid/Missing inputs on Comment creation | Comment information missing fields | Comment error should be displayed | No error is given and "Assigned Tech: undefined (undefined)" added to ticket info | No |
| 7.1 | Closing Ticket | Close ticket button clicked | Ticket should have status changed to closed | Ticket status shows closed | Yes |
| 7.1 | Re-opening ticket | Re-open ticket button clicked | Ticket should have status changed to open | Ticket status shows opened | Yes |
| 7.2 | Assigning ticket | Assign ticket button clicked | Ticket should be assigned to tech | Ticket now shows tech assigned | Yes |

* 1. **Suspension criteria**

If there are over 30% of tests failed, the testing is suspended until the development team fixes all the failed test cases.

* 1. **Exit criteria**

Run rate = number of test cases executed / total test cases. Must be 100% unless a clear reason is given.

Pass rate = number of test cases passed / total test cases. Must be 95% with 5% of not crucial defects.

The Project Manager is to confirm that testing is successfully completed.

* 1. **Test Deliverables**

Screenshot deliverables are attached to a file of **Test Case Screenshots** for the processes that were done for the test cases

1. **Design**
   1. **Introduction**

**Purpose**

This design will detail the implementation of the requirements as defined in the Software Requirements Specification

**System overview**

The project is comprised of two users, an end user and a help desk support tech. An end user would be able to submit tickets, keep track of tickets, and send messages to the techs who have their ticket. A help desk support tech would be able to assign themselves a ticket, send messages to the end user, and change the status of the ticket.

* 1. **Design Considerations**

**Assumptions and Constrains**

Users will enter tickets in when they are having issues, and tickets will be assigned to a tech.

**System Environment**

*Hardware*:

Web Server: AWS EC2 virtual server on the t2.micro tier

Database Server: AWS RDS virtual db on the db.t3.mucro tier

*Performance:*

Web Server: 1 vCPU @ 2.5 GHz, 1GB memory, 8GB storage.

Database Server: 2 vCPU @ 2.5 GHz, 1 GB memory, 20 GB storage.

Data base is Postgres.

* 1. **Design Methodology**

Agile method is being used. Iterative development is chosen because requirements and solutions evolve through collaboration between cross-functional team members. The ultimate value in Agile development is that it enables team to deliver value faster, and the schedule for the project is tight.

*Risks and Volatile Areas:* None have been identified.

* 1. **Architecture**

The workflow for the tickets is as follows:

logging in -> building ticket -> working on ticket -> finishing of the ticket.

Components of the system are ticket board managing tickets and a ticket which holds the information of the ticket, opening a ticket.

The functional decomposition is broken out into what each type of user is responsible for. The users and their responsibilities are as follows:

|  |  |  |
| --- | --- | --- |
| *User* | *Tech* | *Admin* |
| Opening Ticket  Interacting with Tech  Closing Ticket | Opening Ticket for users  Interacting with End-User  Managing Ticket variables  Closing Tickets | Managing Users |

* 1. **Systems and components**

*Ticket Board:*Showing tickets of respective users.

Being able to manage tickets and sort through them

Showing information of tickets, status, user, last updated, ticket number, assigned

Flow into looking at an individual ticket

Flow into opening a ticket

Search through ticket directory

*Ticket Info:*  
Being able to add comments to a ticket

Being able to adjust status of ticket

Retrieving information of ticket

Opening a ticket:

Add initial description of ticket

Add user to ticket

Set information to default of a newly open ticket

* 1. **Strategy**

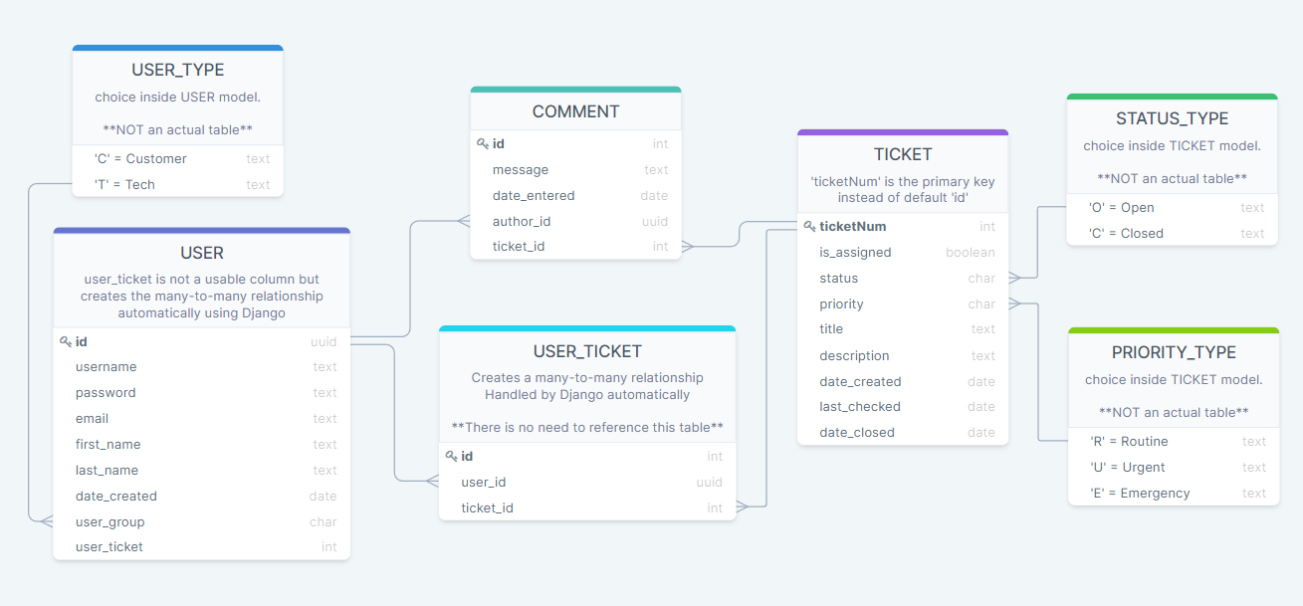
The two design strategies that were used were a structured design and a bottom-up design.

For a structured design the modules were broken out in multiple ways to handle it and how to interact with each other. Building a ticket, how to show the tickets, and interacting with the ticket was the most important aspects of the design. Then to find out how to have these modules interact with each other is the next step.

For a bottom-up design the most important part of the project is the ticket, how a ticket is managed and interacted with dictates how the project is being built. This strategy was chosen because of the back-end work of the project.

* 1. **Database Schema**

The Database schema and detailed description of the tables are represented below.



The table on the next page represents the details on tables: fields, data types for the fields, and fields description.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Field Name** | **Data Type** | **Allow Nulls** | **Field Description** |
| User | id | UUID4 |  | primary key, auto generated |
|  | username | CharField(32) |  |  |
|  | password | CharField(32) | Yes |  |
|  | email | EmailField |  | unique |
|  | first\_name | CharField(20) |  |  |
|  | last\_name | CharField(24) |  |  |
|  | date\_created | DateTimeField |  | auto generated |
|  | user\_group | CharField(1) |  | choice ('C', 'Customer'),  ('T', 'Tech') |
|  | user\_ticket | ManytoMany |  | Many to Many relationship to Ticket table using Django. Uses an association class table called user\_ticket to handle automatically.  This is not a useable column in the actual database. |
|  |  |  |  |  |
| Ticket | ticketNum | AutoField |  | primary key, auto created, used instead of default ‘id’ primary key |
|  | is\_assigned | BooleanField |  | default is ‘False’ |
|  | status | CharField(1) |  | choice ('O', 'Open'),  ('C', 'Closed')  default is ‘Open’ |
|  | priority | CharField(1) |  | choice ('R', 'Routine'),  ('U', 'Urgent'),  ('E', 'Emergency') |
|  | title | CharField(50) |  |  |
|  | description | TextField |  |  |
|  | date\_created | DateTimeField |  | auto generated |
|  | last\_checked | DateTimeField | Yes |  |
|  | date\_closed | DateTimeField | Yes |  |
| **Table Name** | **Field Name** | **Data Type** | **Allow Nulls** | **Field Description** |
| Comment | id | AutoField |  | primary key, auto generated |
|  | message | TextField |  |  |
|  | date\_entered | DateTimeField |  | auto generated |
|  | author\_id | ForeignKey |  | Many to One relation to User table  on\_delete is set to ‘Protect’ |
|  | ticket\_id | ForeignKey |  | Many to One relation to Ticket table  on\_delete is set to ‘Protect’ |
|  |  |  |  |  |
| User\_Ticket | id | AutoField |  | primary key, auto generated |
|  | user\_id | UUID4 |  | Many to One relation to User table |
|  | ticket\_id | Int |  | Many to One relation to Ticket table |

User\_type, status\_type, and priortiy\_type are kept as a list of choices inside their parent table instead of their own table.  For the scope of this project and no plans for future scaling this simplifies things for the backend development when sorting these tables.

The user\_ticket table is created and managed automatically by Django when the many-to-many relation is created under the user\_ticket column in the table user.

Tickets will have 1 'Customer' user and at least 1 'Tech' user.

* 1. **High Level and User Interface Design**

The user will need to login to view the tickets. User can sort existing tickets, comment on a ticket, or create new tickets.

To create ticket user needs to enter summary and description, choose priority and press the “Submit” button. Only tech user has option “Create User for Ticket”.

Graphical user interface, application

Description automatically generated

To view the ticket information the user needs to click on the ticket description text when on the main page.

Graphical user interface, application

Description automatically generated

The “Ticket Information” shows the detailed information on the ticket. The possibilities to add comments (for both, tech and end user) or change status and assign ticket (for tech user only) are also represented on this page.

Graphical user interface, application

Description automatically generated

Added comments will appear on the “Ticket Information” area. To add comment need to click the corresponding link on the bottom.

To sort the ticket, users need to choose criteria. If several criteria are chosen, result will filter satisfy all of them.

Graphical user interface, application, Word

Description automatically generated

At the end of the session user should logout by pressing the link in the header.

* 1. **Low Level Design**

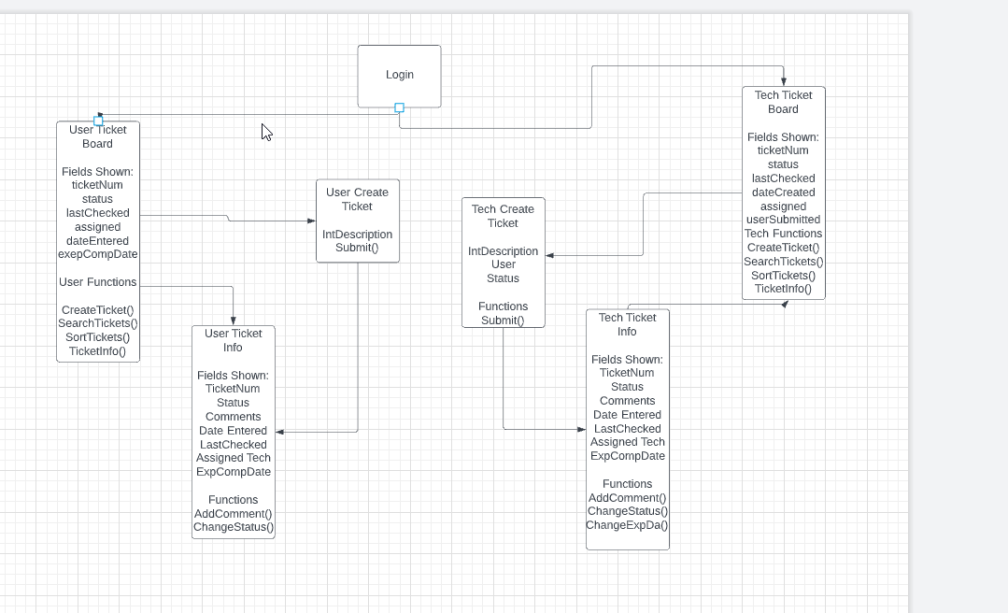
Data structures will be database tables.

Inputs will be handled by the POST method for forms.

Outputs will be handled by html templates using tables.

Performance issues are not identified.

Fields and functions are represented on the diagram on the next page.



**7.10 Alternate designs**

* Considered having a top-down design on the data base vs a bottom-up for managing tickets because of how the database will be queried with how to find tickets by user, we decided to go with a bottom-up
* Adjusted the project file layout with having a split on views and models vs having them all in one file, having them separated made it easier for adjustments as a group
* Adjusted the design slightly to have the buttons on the home page to have them segregated a bit

1. **Development History**

*Before the development*

The overall project history starts with the idea, that appeared to be interesting enough and was approved by professor. General description was represented in the week-1 discussion forum.

Product Specifications (Requirements Specifications) and Project Plan were set to specify in more details what, when and by who should be done. Functionality of the product and timeline were set for the future work.

User’s Guide and Test Plan were completed to have a better idea of the final user interface and user scenarios of the program. The criteria of successful implementation of all functionalities were set in the test plan.

Product Design was completed to show the technical details of the product.

Documentation - Karina D'Abbraccio with assistance of other on the technical details.

*Development*

The GitHub repository was set up for the project to allow the teamwork and control. The work on the project started earlier, which allowed additional time for the three weeks of development that are allocated by the class schedule.

|  |  |  |
| --- | --- | --- |
| **Planned** | **Accomplished** | **Responsible person** |
| **Phase – I** | | |
| Assisting with coding  Assisting with documents | Assisting with coding  Assisting with documents | Dalton Coughlin |
| Deployed AWS server  Established Github for team  Starter Django project configured with PostgreSQL on server | Deployed AWS server  Established Github for team  Starter Django project configured with PostgreSQL on server | Kobi Hsu |
| Assisting with backend work for ticket views | Assisting with backend work for ticket views | James Sweatt |
| Assisted with designing DB  Assisted with AWS Server  AWS DB Server created and running Postgres  DB built using Django models - some changes requested | Assisted with designing DB  Assisted with AWS Server  AWS DB Server created and running Postgres  DB built using Django models - some changes requested | Jared Rice |
| Assisting with CSS | Assisting with CSS | Karina D'Abbraccio |
| Built default templates for project  Assisting with CSS for project  Assisting with backend for project | Built default templates for project  Assisting with CSS for project  Assisting with backend for project | Grant Collins |
| **Status of Project:**   * Login for users are working * Home page for tickets base functionality has been set up * Creating new tickets base functionality has been setup * AWS Server setup and project on the server * Tech docs have been completed | | |
| **Phase – II** | | |
| Assisting with coding  Assisting with documents | Assisting with coding  Assisting with documents | Dalton Coughlin |
| Work on user sessions | Finished the sort functionality on the home page;  Started on unit testing. | Kobi Hsu |
| Assist with backend work for individual ticket views Rework user model for ticket management | Incorporated Django's default user model into the project;  created sample data used for development;  added 'comments' to ticketinfo page;  added 'submitted by' to ticketinfo page;  added 'assigned techs' to ticketinfo page;  'last\_checked' field is updated when a tech views a ticket;  auto assign user to ticket when ticket is created;  assisting with database setup and consulting. | Jared Rice |
| Working on home page for ticket retrieval by user and front end work for searching tickets | searchticket.py: added 'allticket' and 'myticket' functionality;  homefunc.js: added 'allticket', 'myticket', 'buildtable' functions;  base.html, home.html: adjusted MyTickets/AllTickets onclick, 'home' link. | James Sweatt |
| Update documents  Assist with CSS  Starting to work on front-end development | CSS for the home page  Updated documents. | Karina D'Abbraccio |
| Work on front end for individual ticket view  Work on front end redesign for creating ticket | research CSS;  CSS and html for the ticket view. | Grant Collins |
| **Status of Project:**   * Home page advanced functionality setup * Create ticket advanced functionality setup * Individual ticket views setup * Database changes made and completed | | |
| **Phase – III** | | |
| Assisting with coding | Assisting with coding | Dalton Coughlin |
| Formatting dates and null values on home page, ticket view;  word wrapping on ticket page;  description and long comments don’t run off the page. | Formatted dates and null values in home page and ticket view;  Worked on word wrapping on ticket page (long descriptions and comments should not run off the page). | Kobi Hsu |
| Finish adding user to ticket if user is tech;  Logout functionality. | Finish adding user to ticket if user is tech;  Logout functionality. | James Sweatt |
| Submitting comments;  Users may only see their own tickets. | Hide 'All Tickers' tab from Users;  Users and Techs can now add comments to tickets;  User can only view and filter their own tickets;  Techs can now assign themselves to a ticket;  Techs can open and close tickets. | Jared Rice |
| Fix design functional and style bugs;  update documents. | Fix design functional and style bugs;  update documents. | Karina D'Abbraccio |
| Style create ticket page. | CSS stylings for the single ticket view. | Grant Collins |
| **Status of Project:** | | |
| All functionality is developed;  Testing in progress. | | |

After the three phases of development, the last stage is Testing of the final product and preparation of the Final report.

GUI has been tested on May, 4: 2 bugs in UI found and fixed by May, 5 (K.D’Abbraccio).

1. **Conclusions**

As was estimated in the Risk Assessment section of the Project Plan, the resource and communication issues appeared during the project, which did not affect the successful completion of the project with all planned functionality present and working correctly in the final product. Therefore, we consider the project has been completed successfully.

There are ways to personally continue working on this project, because the source code is publicly available on GutHub.

The current product may be expanded and *improved* in future by including the following functionality:

- reports or other metrics on tickets;

- create pre-written responses;

- accept messages from different channels;

- provide tutorials or answers on frequently asked questions;

- create and send customer satisfaction surveys;

- customized dashboards. (Karina)

*Design strengths and limitations*

Python Django has a default, built-in database that uses SQLite. Although for many applications, this database suffices, and probably would have for this project. However, SQLite is limited so we decided to use PostgreSQL. Implementing this database design from the beginning can prove to be a significant advantage in the future. If we chose to use the default databasing, by the time the database outgrew it’s capability, it can be very difficult, if even possible, to dump and load into a different database like PostgreSQL. (James)

*Lessons learned (style saved as the author):*

Dalton Coughlin:

Lessons learned:

* How to manage people and set appropriate deadlines
* Walking people through processes

Improvements:

* Improved ticket search
* Pagination on home page
* More granular data for tickets

Limitations:

* Issues with finding tickets when ticket base gets large
* Issues with comments getting unreadable after comment base gets large on ticket
* Management of users has to be done through Django admin, should of built in management console of users

Kobi Hsu:

* No comments given

*Karina D’Abbraccio:*

* Working in a team is not new to me, so the workflow advantages and disadvantages are more or less similar to other groups I have participated in; the difference is that the PM in the school project doesn’t have the authority to take action like in the workplace;
* Gained basic skills in using version control system (Git, GitHub), participated in the group project that is now reflected in my GitHub account;
* Learned basics of Django and have a goal to develop a small personal project later;
* Learned basics of database development and get recommendations for the future learning;
* Worked with the code written by other students;
* Was able to see projects and other tips from more experienced team members for the future career.

*Jared Rice:*

Lessons Learned:

This class projected is used to simulate a real-world working environment where an application is developed and built from the bottom up with a team. Instead of jumping right into coding, we took the first few weeks to build a team, plan a project, develop a test plan, and design the application. I believe without these critical steps; most projects would fail to reach production due to poor planning. This type of project planning will follow from simple side projects to future software development jobs.

Another lessoned learned is not being tied to a specific set of languages or stack. Until this class I had never used javascript, html or Django and very minimal experience with python. I was able to meet the requirements set by my “project manager” for setting up the database then reached out and assisted with other portions of the project. This built confidence in learning new languages and libraries rapidly in order to meet the needs of the team.

Lastly, I would say communication skills are key in any team environment. I think clearly defining task and goals among members avoids any confusion and delays. These expectations should also be discussed and reevaluated if issues arise. One issue I personally ran into was not knowing how to connect the front end and back end. I was given time and good sources for review. Another issue was losing power and internet for 5 days due to a construction mishap nearby. Without effectively communicating to the project manager, this could have caused delays in our project.

Improvements:

Python Django has a default, built-in database that uses SQLite. Although for many applications, this database suffices, and probably would have for this project. However, SQLite is limited so we decided to use PostgreSQL. Implementing this database design from the beginning can prove to be a significant advantage in the future. If we chose to use the default databasing, by the time the database outgrew it’s capability, it can be very difficult, if even possible, to dump and load into a different database like PostgreSQL.

*Grant Collins:*

*Lessons learned:*

* It is best to complete most of the planning before any coding takes place
* Be sure to communicate any impediments to the Project Manager as soon as possible

Improvements:

* Set aside more time to work on coding for the project
* Communicate roadblocks earlier

Limitations:

* Skills/knowledge gap with programming languages and frameworks
* Time constraints on each deliverable