Group 2

User Accounts information:

The application uses a predefined set of accounts. Some of them are listed below but the frequently used account for testing is “test” pass: “test”. The accounts with status “Admin” can be used to test admin level features of the application.



Lorenzo Krol 629808

**What did I do?**

I did the DAO and service for the login. I also made the user management part with added functions like editing and deleting users.

**DAO and service for the Login**

For the database interactions I used service and DAO layers, because it clean and the database interactions are separated. For retrieving user information from the database I used class casting so that retrieving and filling a class is quicker.

**User management panel**

Initial user management panel

I made a list view that displays all the users and corresponding information. I used a user list for retrieving all the users, because lists will scale with the amount of users in the database and used that list to fill the list view. There is a filter by email text box that changes the info in the list view based on what the user types in that textbox. When the user enters something in the textbox, a method will be called that is almost the same as the normal fill list view method, except that this method uses an if statement that checks if the email contains the user input. On the initial panel there is also a delete button. When pressed, the method behind the button will first check if a user in the list view is selected. If that is not the case it will show a message box with the question to select a user. When a user is selected and the delete button is clicked again, it will delete the user in the database with the email as filter so the database knows which user to delete.

Edit/Add button and panel

When this button is pressed, a check will be done to see if a user in the list view is selected and a panel with two buttons appears. Add and Edit. If a user is selected, then the add button will become unavailable, textboxes will be filled with existing user data and if there is no user selected, the edit button becomes unavailable and the textboxes are empty. This is because the edit and add buttons use the same textboxes. In the panel there is a hidden label which stores the original email in case the user email changes. This hidden label is not being used for adding a user. The user password is retrieved from the database, because the list view does not save passwords. All the textbox info will be send to the user DAO through the user service and based on the users steps, a user will be added or changed. For adding a user, an authorization code will be set to 0 and saved to the database. This is done so that the order of the info in the NoSQL database will correspond with the order of info in the user class, otherwise the class casting can go wrong and the order wont be correct when retrieving and casting it into a class.

Max van Kekeren 629466

**I forgot my password**

I made the “I forgot my password” function. Here the user enters his or her email. The program checks if that email exists in the database and generates an AuthCode. The Authcode is emailed to the user. The user is asked to enter the authcode into NoDesk and NoDesk verifies of the Authcode is correct. After verification the user is able to change their password.

**Change ticket status**

When opening a ticket the user has the possibility to change a ticket status. This status will be updated in the database. After the change is updated in the database the user receives an email regarding their ticket.

**User management Notifications**

When an Admin Adds, edits or deletes a user within the user management segment the user receives an email regarding the change. When you are added you’ll receive a welcome message.

**Error handling**

Through-out the application you’ll find various error handling methods.

Muskan Bhat 636130

**Login System Form**

I made a Login form for the application. The user enter his/her email and password in the form and if the email/username or password does not belong to any registered account in the database, the application displays and error saying login details are incorrect. The user accounts are manually registered in the database to test the application username: “test” and password: “test” can be used.

**Incidents Sort**

On the list of all the incidents in the incident management, I made it possible to sort the incidents on the basis of specific columns. The indents can be sorted in ascending and descending order if the user clicks any column header of the list. I made it using “ListViewColumnSorter” using the interface “IComparer”.

**Filter by Email**

The list of all incidents can be filtered on the basis of user email on the list. If the user inputs keywords that the user email contains, the list will be filtered in real time with the incidents related to the keyword in the text box.

**Exporting Incidents List to .csv file**

In the incident management, below the list of incidents a button “export incidents list to .csv file” is provided which when clicked the csv in the system. The file is aimed to be saved in a sub-folder “Incidents Lists” under the special folder “documents/ my documents”. A function first check if the subfolder “Incidents Lists” already exists in the system, and if not it creates the respective sub-folder. Lastly, it displays a message confirming the file is saved and also displays the location of the file.

Ivaylo Nachev 627341

**What did I do?**

I was responsible for implementing the dashboard, creating an incident ticket menu and ticket overview based on selected item by the user.

**Dashboard**

Created initially the ticket\_DAO class to serve the purpose to interact with the database when handling tickets.db\_TicketsList and GetALLTicketsList methods retrieve all tickets in the database and hand them over to the service layer class. In service layer method GetAllTickets was created to retrieve a list of tickets from database and return it as list.Then also made the IncidentCount to count the total amount of incidents(solved or pending), UnresolvedCount for the unresolved and PastDeadlineCount for the unresolved tickets that are past their deadline.I Implemented the circular progress bars by installing NuGet package to fit the visual design requested by the client. Created a Ticket class in Model with private variables and only getters when said variables must be read from outside and variables can be written only in the constructor for security reasons. Also created a method that checks if a ticket is past its deadline and still unresolved and sets its status to PastDeadline. Enumerations Status and Priority were also made to avoid passing strings between layers and avoid making mistakes by doing so and to make code more readable. Show List button on the dashboard was decided by the entire team to lead to the incident management panel where users can see all unresolved tickets and avoid doing the same thing all over again.

**Create a ticket**

In DAL layer created a method DB\_Write\_ticket to write information into the database from a ticket by user input. Made service layer method WriteTicket\_service that hands over ticket information from View layer to database layer for writing and assign new ID by calling GetId method from IdCenerator class. IdGenerator class was created to avoid possible null error for ID in MongoDB even if MongoDB usually alway automatically assigns object id by itself. Onto ticket creation panel a clock was implemented to quickly orient users of the current time and date. Text fields were implemented for direct user input by typing for requested from, subject, description and type of incident. For choosing a priority it was implemented based on the design provided from the client. Lastly instead of combobox a datepicker was implemented to provide more flexibility and intuitiveness for users when they select a deadline for an incident ticket.

**Ticket overview (special feature)**

Made a new panel which is shown to the user after selecting a solved or unresolved incident from their representative list Views and then clicking the “open incident ticket” button. On this panel all ticket information is displayed to the user and also provides a clock that provides the user with the current data and time.