**CHANGE REQUEST MANAGEMENT APPLICATION**

**Graduation Project Report**

**CMSE 406**

**Team members:** Talal Mahdy – 147139

Mohamed Balto - 147697

Adham Moshasha - 148387

Abdoulgwad Elsheredi -147597

**Supervisor: Assist. Prof. Dr. Adnan Acan**

**Computer Engineering Department**

**Eastern Mediterranean University**

**Spring 2018**

# ABSTRACT

The aim of this project is to produce a Web Application to manage and control and various types of requests at the Kuzey Kibris Turkcell telecommunications company. The application’s main objectives are to manage, organize and track the many requests from the employees of all departments of KKTCell while effectively communicating the changes to the managers of those departments. This project was started mainly to reduce the paperwork, effectively handle any request related information, and to improve the communication among the team members of KKTCell. The Web Application will be developed using the ASP.NET framework along with C# as a backend language. As a result, the final output of the project is a well-designed, well-documented, easily accessible and easy to use Web Application known as the Change Request Management Application.

**Keywords:** Web Application, Change Requests, Organization, Employees, ASP.NET MVC**Table of Contents**

[ABSTRACT II](#_Toc515887492)

[LIST OF FIGURES VI](#_Toc515887493)

[LIST OF TABLES IX](#_Toc515887494)

[1. INTRODUCTION 1](#_Toc515887495)

[2. PROJECT PLANNING AND MANAGEMENT 2](#_Toc515887496)

[2.1. Project Team 2](#_Toc515887497)

[2.2. Organization Scheme 3](#_Toc515887498)

[2.3. Tools/Methods Applied 3](#_Toc515887499)

[2.4. Reason for starting the Project 3](#_Toc515887500)

[2.5. Success Criteria 4](#_Toc515887501)

[2.6. Software Development Plan 4](#_Toc515887502)

[2.7. Software Cost Estimation - COCOMO 5](#_Toc515887503)

[2.8. Work Packages and Gantt Chart 6](#_Toc515887504)

[2.9. List of Milestones 19](#_Toc515887505)

[2.10. List of Risks 19](#_Toc515887506)

[2.11. Commercialization Potential 20](#_Toc515887507)

[2.12. Project Economic Expectations 20](#_Toc515887508)

[2.13. Software Purchases 21](#_Toc515887509)

[2.14. Quarterly Estimated Cost Form (TL) 23](#_Toc515887510)

[3. REQUIREMENTS ANALYSIS 24](#_Toc515887511)

[3.1. Functional Requirements 24](#_Toc515887512)

[3.2. Non-Functional Requirements 32](#_Toc515887513)

[3.2.1. Software Quality Attributes 32](#_Toc515887514)

[3.2.2. Performance Requirements 32](#_Toc515887515)

[3.2.3. Safety Requirements 32](#_Toc515887516)

[3.2.4. Implementation Requirements 32](#_Toc515887517)

[3.2.5. Security Requirements 32](#_Toc515887518)

[3.3 Ethical issues 32](#_Toc515887519)

[4. DESIGN 33](#_Toc515887520)

[4.1. High level design 33](#_Toc515887521)

[4.1.1. System Architecture - MVC 33](#_Toc515887522)

[4.1.2. Network Architecture 34](#_Toc515887523)

[4.1.3. Context Diagram 35](#_Toc515887524)

[4.1.4. Dataflow Diagram – Level 0 36](#_Toc515887525)

[4.1.5. User Interface Design 37](#_Toc515887526)

[4.2. Low level design 46](#_Toc515887527)

[4.2.1. Dataflow Diagram – Level 1 46](#_Toc515887528)

[4.2.2. Data Dependencies 47](#_Toc515887529)

[4.2.3. Class Diagram 48](#_Toc515887530)

[4.2.4. UML Interaction Diagrams 49](#_Toc515887531)

[5. IMPLEMENTATION 53](#_Toc515887532)

[5.1. Tools, technologies, platforms, and libraries used 53](#_Toc515887533)

[5.2. Use of Software Engineering Process Steps 53](#_Toc515887534)

[5.3. Algorithms 55](#_Toc515887535)

[5.4. Standards 57](#_Toc515887536)

[5.5. Detailed description of the implementation 58](#_Toc515887537)

[6. TESTING 67](#_Toc515887538)

[7. USER GUIDE OF THE SYSTEM 71](#_Toc515887539)

[8. DISCUSSION 80](#_Toc515887546)

[9. CONCLUSION 81](#_Toc515887547)

[10. REFERENCES 82](#_Toc515887548)

[APPENDIX A 84](#_Toc515887549)

[APPENDIX B 85](#_Toc515887550)

[APPENDIX C 98](#_Toc515887551)

# LIST OF FIGURES

Fig 1. Organization Scheme.......................................................................................................3

Fig 2. Gantt Chart (i)................................................................................................................11

Fig 3. Gantt Chart (ii)...............................................................................................................12

Fig 4. MVC Architecture.........................................................................................................33

Fig 5. Client-Server Architecture.............................................................................................34

Fig 6. Context Diagram............................................................................................................35

Fig 7. Level 0 Data Flow Diagram...........................................................................................36

Fig 8. Login Page Design.........................................................................................................37

Fig 9. Home Page Design.........................................................................................................38

Fig 10. New Request Page Design...........................................................................................39

Fig 11. Panel 1 Design.............................................................................................................40

Fig 12. Panel 2 Design.............................................................................................................41

Fig 13. Panel 3 Design.............................................................................................................42

Fig 14. Panel 5 Design – Add New SMS.................................................................................43

Fig 15. Panel 5 Design – Import SMS....................................................................................44

Fig 16. Panel 8 Design.............................................................................................................45

Fig 17. Level 1 Data Flow Diagram (Form Updating).............................................................46

Fig 18. Entity Relation Diagram..............................................................................................47

Fig 19. Class Diagram..............................................................................................................48

Fig 20. Use Case Diagram........................................................................................................49

Fig 21. Sequence Diagram – Login to System.........................................................................50

Fig 22. Activity Diagram – View Request...............................................................................51

Fig 23. Business Process Model...............................................................................................52

Fig 24. Incremental Method of Development..........................................................................53

Fig 25. React Files frontend Implementation...........................................................................59

Fig 26. Error 404 Implementation............................................................................................60

Fig 27. Model Classes..............................................................................................................62

Fig 28. Employee.cs business logic Class................................................................................62

Fig 29. NewFrd.cs Presentation logic ViewModel Class.........................................................63

Fig 30. DB\_Functions.cs Database Implementation Class......................................................63

Fig 31. RequestController.cs Controller Class.........................................................................64

Fig 32. FRD\_ACTIVATED Trigger function in Database......................................................65

Fig 33. Database Tables in Oracle............................................................................................65

Fig 34. Login functions Functions.cs and LogedInEmployee.cs.............................................66

Fig 35. Login Page...................................................................................................................73

Fig 36. Main Landing Page......................................................................................................73

Fig 37. Create New Request – Panel Page...............................................................................74

Fig 38. Panel 0 and Panel 1......................................................................................................74

Fig 39. Panel 2..........................................................................................................................75

Fig 40. Panel 3..........................................................................................................................75

Fig 41. Panel 4..........................................................................................................................76

Fig 42. Panel 5 and Panel 6......................................................................................................76

Fig 43. Panel 7..........................................................................................................................77

Fig 44. FRD Submitted............................................................................................................77

Fig 45. FRD Status: Pending....................................................................................................78

Fig 46. Manager received FRD................................................................................................78

Fig 47. Manager approves FRD...............................................................................................79

Fig 48. User\_Fikri Collaborating after approval......................................................................79

# 

# LIST OF TABLES

Table 1. Project Team Members................................................................................................2

Table 2. UAF Table....................................................................................................................5

Table 3. Work Package 1...........................................................................................................6

Table 4. Work Package 2...........................................................................................................7

Table 5. Work Package 3...........................................................................................................8

Table 6. Work Package 4...........................................................................................................9

Table 7. Work Package 5.........................................................................................................10

Table 8. Work Breakdown Structure........................................................................................18

Table 9. List of Milestones.......................................................................................................19

Table 10. List of Risks.............................................................................................................19

Table 11. Project Economic Expectations................................................................................20

Table 12. Software Purchases..................................................................................................21

Table 13. Quarterly Estimated Cost Form................................................................................23

Table 14. Test Case 1...............................................................................................................67

Table 15. Test Case 2...............................................................................................................68

Table 16. Test Case 3...............................................................................................................68

Table 17. Test Case 4...............................................................................................................69

Table 18. Test Case 5...............................................................................................................70

Table 19. Test Case 6...............................................................................................................70

# 1. INTRODUCTION

In this project, we should develop a fully functional, well-designed and a well-documented Web Application known as the Change Request Management Application for the Kuzey Kibris Turkcell1 telecommunications company in a period of around 8 months. To do this, we used the Incremental Model of development which is a flexible model for small-medium projects. The main people who will benefit from this project are the employees and managers in the KKTCell company workplace who are going to find it much easier to manage and change the various proposed requests. The application is able to manage any arising request using a special FRD form type in addition to various preset requests such as setting up a new campaign advertisement, changing the SMS contents of a package, adding new discounts to various packages, etc. To prepare this report, a Software Requirements Specification Document2 (SRS) in addition to a Software Design Specification Document3 were prepared. In this report, we will go through all the phases that lead to the building of this project. First of all, the specific requirements of the system will be described. Then, the design of various parts of the system will be shown. After that, the implementation of the system will be explained. Finally, the testing phase will be explained and a user guide will also be shown.

# 2. PROJECT PLANNING AND MANAGEMENT

## 2.1. Project Team

Table 1. Project Team Members

|  |  |
| --- | --- |
| **Project No** | 1 |
| **Project Name** | Change Request Management Application |
| **Start Date** | 25-Sep-2017 |
| **End Date** | 02-June-2018 |
| **Time** | 251 Days |

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Leader/Software Architect/Software Designer/Documenter** | | | |
| **Name Surname** | Talal Mahdy | **ID No** | 147139 |
| **Address** | Famagusta, North Cyprus | | |
| **Phone** | +90 533 8885729 | | |
| **Email** | talal.mahdy96@gmail.com | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Lead Programmer/Web Service Developer/Administrator** | | | |
| **Name Surname** | Mohamed Balto | **ID No** | 147697 |
| **Address** | Famagusta, North Cyprus | | |
| **Phone** | +90 533 8397554 | | |
| **Email** | baltu.libya@gmail.com | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Software Programmer/Software Tester/Maintainer** | | | |
| **Name Surname** | Abdoulgwad Hussien Elsheredi | **ID No** | 147597 |
| **Address** | Famagusta, North Cyprus | | |
| **Phone** | +90 533 8528065 | | |
| **Email** | abdoulgwad.elsheredi@yahoo.it | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements Engineer/User Interface Designer/Database Developer** | | | |
| **Name Surname** | Adham Moshasha | **ID No** | 148387 |
| **Address** | Famagusta, North Cyprus | | |
| **Phone** | +90 533 8725650 | | |
| **Email** | adhamoshasha@gmail.com | | |

## 2.2. Organization Scheme

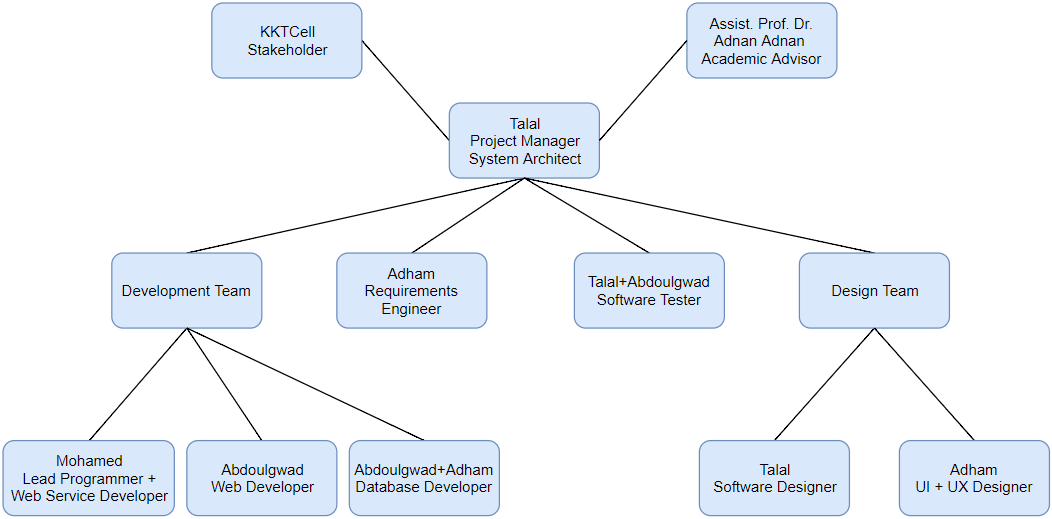


Fig 1. Organization Scheme

## 2.3. Tools/Methods Applied

**Project Management and Scheduling:** Microsoft Office, Microsoft Project4  
**Version Control:** Visual Studio Team Services (VSTS)5   
**Design:** Modelio6, Wireframe.cc7, draw.io8, Bootstrap Studio9  
**Implementation:** HTML, CSS, JavaScript, C# Programming Language, Oracle Database10, Visual Studio IDE11, ASP.NET RESTful Web API, JSON12, Bootstrap13, Sass14, JQuery15, React16, LINQ17, Webpack18, Babel19, List.JS20, ASP.NET MVC, SHA-256 Hashing with Salt21, Google Chrome Developer Tools.

## 2.4. Reason for starting the Project

This project was started due to a request by the KKTCell Company. The project was analyzed for its feasibility and a decision was made by our team to start working on this project. Another reason that made us start this project was due to our desire in improving the overall quality of services at KKTCell which in turn will improve the economy and standard of living in Northern Cyprus. Developing this application could in turn assist in the introduction of some bigger & more organized services or the introduction of a new technology in Northern Cyprus such as 4G LTE or 5G.

2.5. Success Criteria  
Success Criteria are metrics to determine if the project is successful. Some of them are:

1. Total Installations: The number of times the application was installed should be growing at a steady rate.
2. Monthly Average Users (MAU): The application should have a high number of Average users among those who installed the application. If it appears that the MAU is growing, then the project is growing in the right path.
3. Engagement: Also, those users should have a high engagement ratio, i.e., users visit it frequently and use it for a considerable amount of time. Engagement can be measured by metrics such as session length (time period between app open and close), session interval (time between the user’s first session and their next one) and retention rate (users who return to the application based on the date of their first visit).
4. Documentation: The number of users submitting help requests or bug reports should be less due to well documentation good design of application.

2.6. Software Development Plan

In this project, we will be applying an evolutionary development approach. An evolutionary development is based on the idea of developing an initial implementation, exposing it to the customer’s comments, refining it through many versions until an adequate system has been developed. This development method is going to be more effective in this project since it is a small-medium sized system. The requirements for this system are not well defined from the beginning and we have to work with the customer and produce prototypes while obtaining feedbacks from the customer. To do this, we are going to have to conduct a number of interviews with the stakeholders and clearly understand the requirements. The main advantages of using this approach to develop this project are:

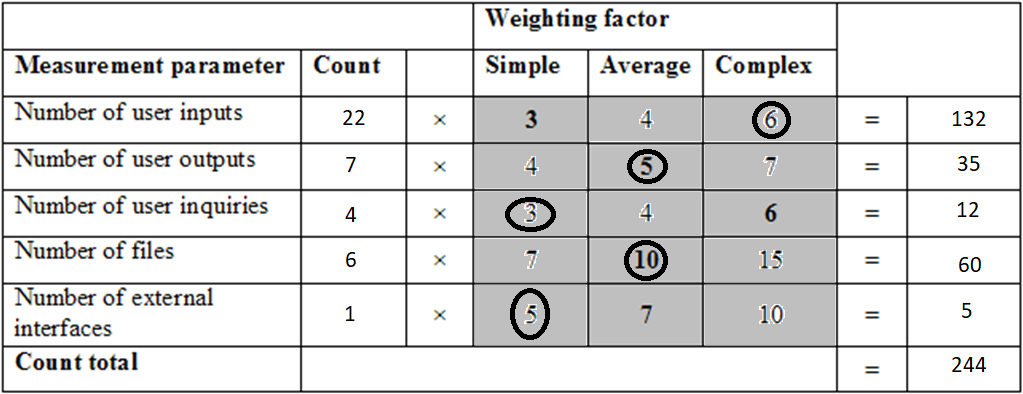
* Each module passes through requirements, design, implementation and testing phases.
* Most important Modules are developed first.
* It is more flexible and less costly to change requirements.
* It is easier to test and debug the modules.
* Delivery of initial Modules is quick.

## 2.7. Software Cost Estimation - COCOMO

We used the Constructive Cost Model22 (COCOMO) to estimate the cost of our project.

Function Point Calculation:

Table 2. UAF Table



Function Points = UFP\*TCF

Function Points = UFP\*[0.65+0.01\*DI]

Unadjusted Function Points (UFP) = 244

Technical Complexity Factor (TCF):

We assume that all Complexity Adjustment Factors are average (Value: 3 on a scale of 5).

DI = summation of all Complexity Adjustment Factors (Total 14) according to influence

Therefore, DI = 3x14=42

Therefore, Function Points = 244\*[0.65+0.01\*42] = 261 Function Points

FP to SLOC Conversion Ratio for C# Language = 29

Therefore, Lines of Code (LOC) = 261 \* 29 = 7569 LOC = 7.569 KLOC

Effort (i.e., man-month) = a\*(KLOC)b = 3.0\*(7.569)1.12 = 29 Man-Month (Semi-Detached)

Duration = c\*(Effort)d = 8.13 Months

Number of people = 20.1/7.81 = 3.57 developers

Total Man-Months = 29 \* 8.13 Months \* 3.57 developers = 842 Hours

## 2.8. Work Packages and Gantt Chart

Table 3. Work Package 1

|  |  |
| --- | --- |
| **Work Package No** | 1 |
| **Work Package Name** | **Feasibility and Pre-Research (SRS stage)** |
| **Start-End Date and Time** | Start: 25-09-17 Finish: 29-10-17 |

|  |
| --- |
| **1- Activities of work package.** |
| 1. **Scope.** 2. **Analysis/Software Requirements.** |
| **2- Methods and parameters that will be used for work package.** |
| Microsoft Project |
| **3- Experiments, tests and analysis in the work package.** |
| 1. **Scope:**    1. Determine project scope    2. Secure project approval    3. Define preliminary resources    4. Secure core resources    5. Scope complete 2. **Analysis/Software Requirements:**    1. Conduct needs analysis    2. Draft preliminary software specifications    3. Develop preliminary budget    4. Review software specifications/budget with team    5. Incorporate feedback on software specifications    6. Develop delivery timeline    7. Obtain approvals to proceed (concept, timeline, budget)    8. Secure required resources    9. Analysis complete |
| **4- Output of work package and its success criteria.** |
| **Outputs:** Initial Requirements Specification Document (SRS), feasibility analysis, secured resources.  **Success Criteria:** Project approved, project is feasible to implement, initial requirements are well documented, resources and team members are secured. |
| **5- Relation of output with other work packages** |
| This is the initial phase of development and is the basic input for all other work packages. It defines the following: What is the project? Who are the stakeholders? Who will use the system? How should it be developed? Who are the team members? What are the basic requirements? How should it be developed? How should it be delivered? Etc. |

Table 4. Work Package 2

|  |  |
| --- | --- |
| **Work Package No** | 2 |
| **Work Package Name** | **System Design (SDS Stage)** |
| **Start-End Date and Time** | Start: 30-10-17 Finish: 13-12-17 |

|  |
| --- |
| **1- Activities of work packages.** |
| 1. **Change Request Management Application Software Design** 2. **Development of first prototype** 3. **Improve SRS Document** |
| **2- Methods and parameters that will be used for work package.** |
| Modelio, Wireframe.cc, Bootstrap Studio, draw.io |
| **3- Experiments, tests and analysis in the work package.** |
| Review preliminary software specifications  Develop functional specifications  Design of System  Develop prototype based on functional specifications  Review functional specifications and Design  Incorporate feedback into functional specifications  Obtain approval to proceed  Design complete |
| **4- Output of work package and its success criteria.** |
| **Outputs:** A Software Design Specification (SDS) Document, First Prototype of Software.  **Success Criteria:** An improvement of the SRS Document as a result of better understanding of requirements from first prototype, completion of system design. |
| **5- Relation of output with other work packages** |
| The design stage is the next stage in the software development life cycle. Without designing the software and knowing what has to be done, it will be very difficult for the programmer to develop the software and many mistakes will be done. So this work package is a very important prerequisite to the next stage which is the development stage. |

Table 5. Work Package 3

|  |  |
| --- | --- |
| **Work Package No** | 3 |
| **Work Package Name** | **Software Development Stage** |
| **Start-End Date and Time** | Start: 12-02-18 Finish: 29-03-18 |

|  |
| --- |
| **1- Activities of work packages.** |
| **The main coding, primary debugging of the program and development of the database.** |
| **2- Methods and parameters that will be used for work package.** |
| HTML, CSS, JavaScript, C#, Oracle Database, Visual Studio IDE, JSON, Bootstrap, Sass, JQuery, React, LINQ, Webpack, Babel, List.JS, ASP.NET MVC + RESTful Web API |
| **3- Experiments, tests and analysis in the work package.** |
| Review functional specifications  Identify modular/tiered design parameters  Assign development staff  Develop Web Service  Develop Database  Develop Application  Developer testing (primary debugging)  Development complete |
| **4- Output of work package and its success criteria.** |
| **Outputs:**  Change Request Management Application  **Success Criteria:**  A successfully functioning Web Application. |
| **5- Relation of output with other work packages** |
| During the development of our application, the coders will obviously find some bugs and attempt to fix them. However, there might be some logical or other types of errors that a developer might not notice. Therefore, it is important for the application to be tested by a separate dedicated tester. Testing of the application can begin shortly after the development of the first unit of the application. |

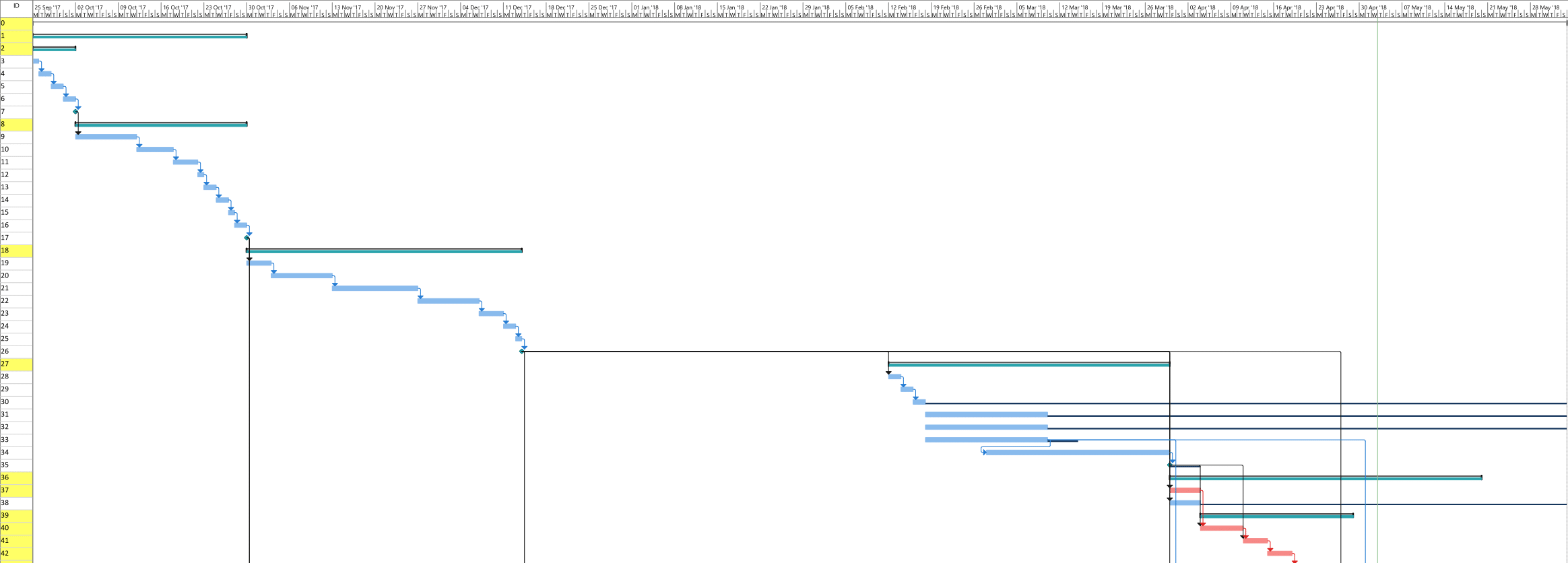
Table 6. Work Package 4

|  |  |
| --- | --- |
| **Work Package No** | 4 |
| **Work Package Name** | **Software Testing Stage** |
| **Start-End Date and Time** | Start: 30-03-18 Finish: 19-05-18 |

|  |
| --- |
| **1- Activities of work packages.** |
| 1. **Unit and Integration Test Plans.** 2. **Unit Testing.** 3. **Integration Testing.** |
| **2- Methods and parameters that will be used for work package.** |
| Module-by-Module Unit Testing and Overall Integration Testing |
| **3- Experiments, tests and analysis in the work package.** |
| 1. **Unit and Integration Test Plans:**     1. Develop unit test plans using product specifications    2. Develop integration test plans using product specifications 2. **Unit Testing:**    1. Review modular code    2. Test component modules to product specifications    3. Identify anomalies to product specifications    4. Modify code    5. Re-test modified code    6. Unit testing complete 3. **Integration Testing:**     1. Test module integration    2. Identify anomalies to specifications    3. Modify code    4. Re-test modified code    5. Integration testing complete |
| **4- Output of work package and its success criteria.** |
| **Outputs:** Test data, verification results  **Success Criteria:** Testing successfully completed with all the errors and bugs successfully fixed. |
| **5- Relation of output with other work packages** |
| After successfully testing the system, next stages in the software life cycle are the delivery and maintenance stages. The software should be delivered and installed as per the request of the customer. Also, the maintenance stage is very important as software may serve for many years to come and it will obviously need to be updated. Therefore, a good maintenance team along with good documentation is very important for the product to be successful. |

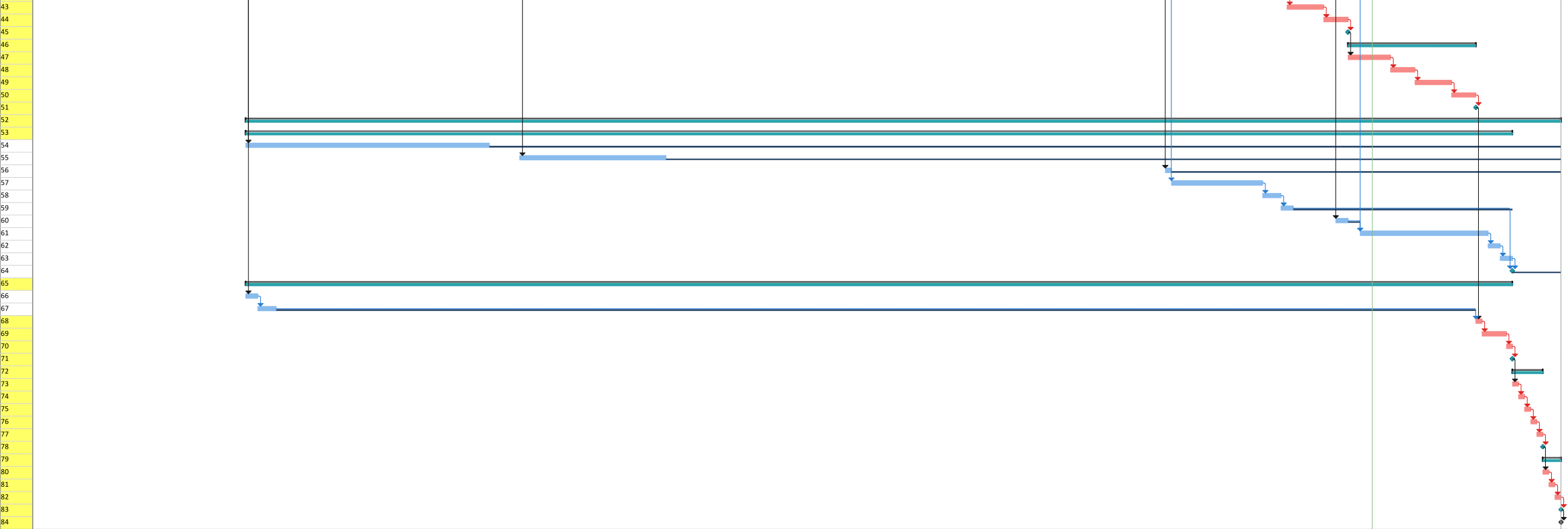
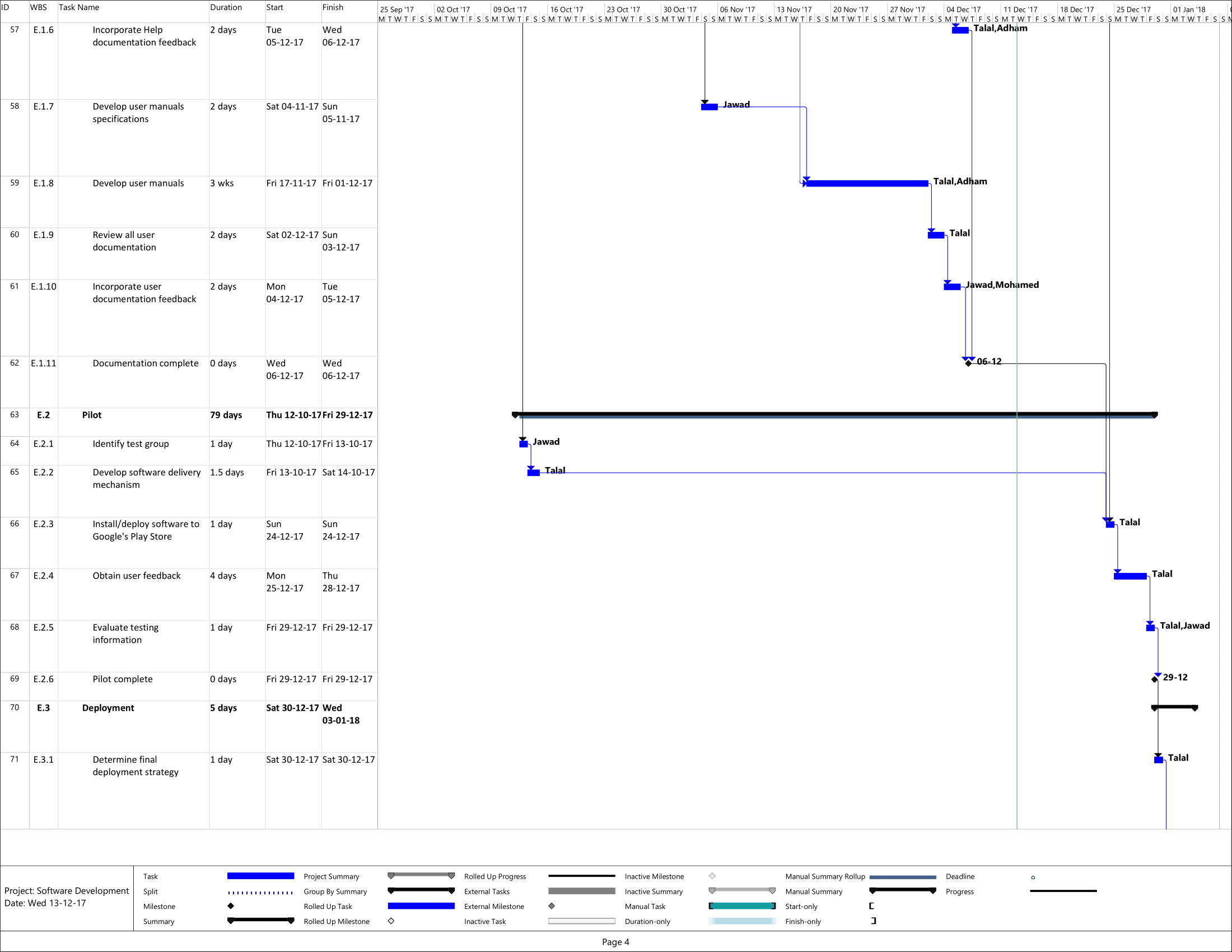
|  |  |
| --- | --- |
| **Work Package No**  Table 7. Work Package 5 | 5 |
| **Work Package Name** | **Documentation and Delivery** |
| **Start-End Date and Time** | Start: 30-10-18 Finish: 02/06/18 |

|  |
| --- |
| **1- List the activities of work packages.** |
| 1. **Documentation** 2. **Pilot** 3. **Deployment** 4. **Post Implementation Review** |
| **2- Methods and parameters that used for work package.** |
| Microsoft Project, Microsoft Office, Visual Studio Team Services, Slack |
| **3- Experiments, tests and analysis in the work package.** |
| 1. **Documentation**    1. Develop Help specification    2. Develop SRS Document    3. Develop SDS Document    4. Develop Help system    5. Review Help documentation    6. Incorporate Help documentation feedback    7. Develop user manuals specifications    8. Develop user manuals    9. Review all user documentation    10. Incorporate user documentation feedback    11. Documentation complete 2. **Pilot**    1. Identify test group    2. Develop software delivery mechanism    3. Install/deploy software to Google's Play Store    4. Obtain user feedback    5. Evaluate testing information    6. Pilot complete 3. **Deployment**    1. Determine final deployment strategy    2. Develop deployment methodology    3. Secure deployment resources    4. Train support staff    5. Deploy software    6. Deployment complete 4. **Post Implementation Review**    1. Document lessons learned    2. Distribute to team members    3. Create software maintenance team    4. Post implementation review complete |
| **4- Output of work package and its success criteria.** |
| **Outputs:** Successful delivery of project, uploading to Play Store, completed documentation  **Success Criteria:** A completed well documented, well perceived software application. |
| **5- Relation of output with other work packages** |
| As can be noticed, the documentation stage started at an early time in the software process, sometime after the design stage started. It is important to document all requirements and design aspects of the project along with a proper user guide before delivering the application. |



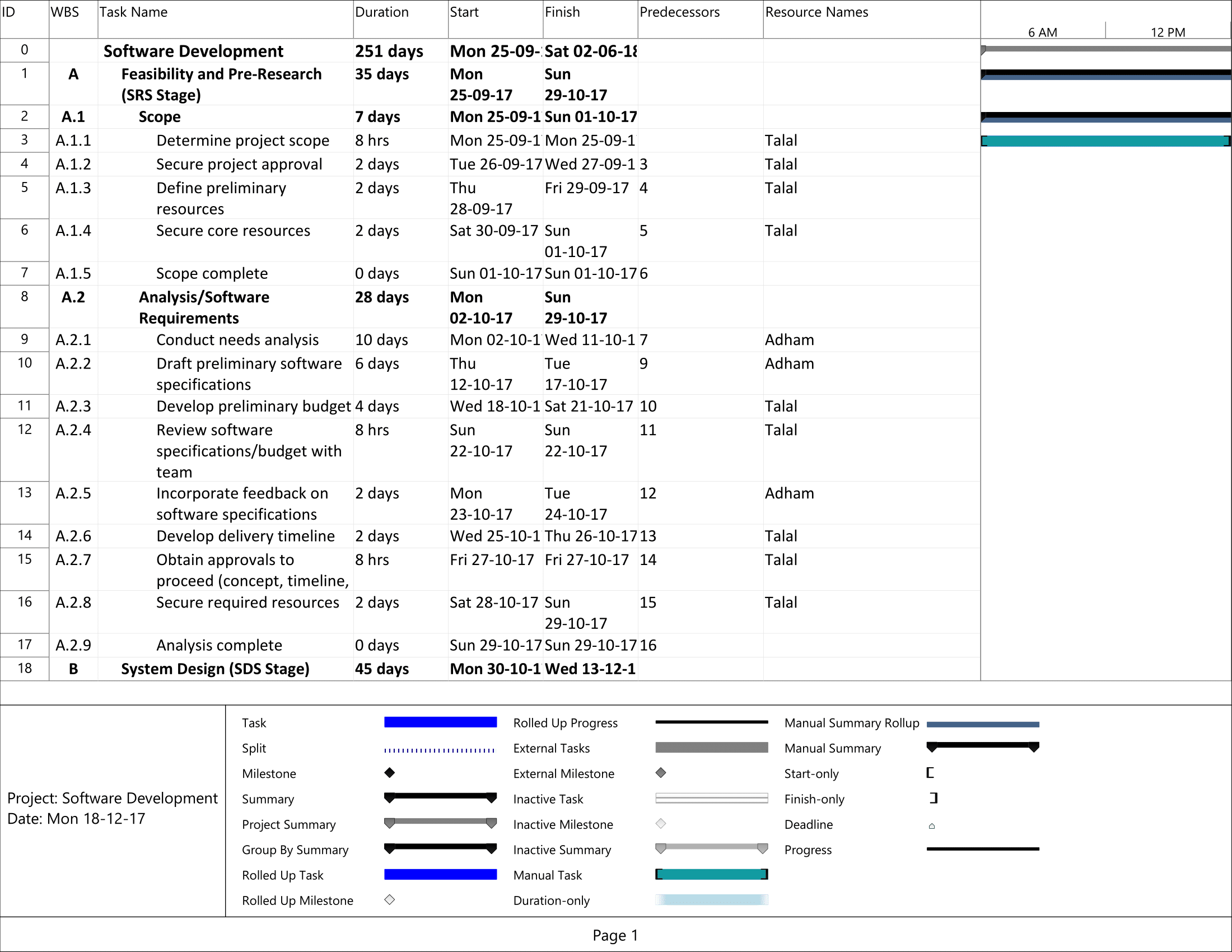
legend footer

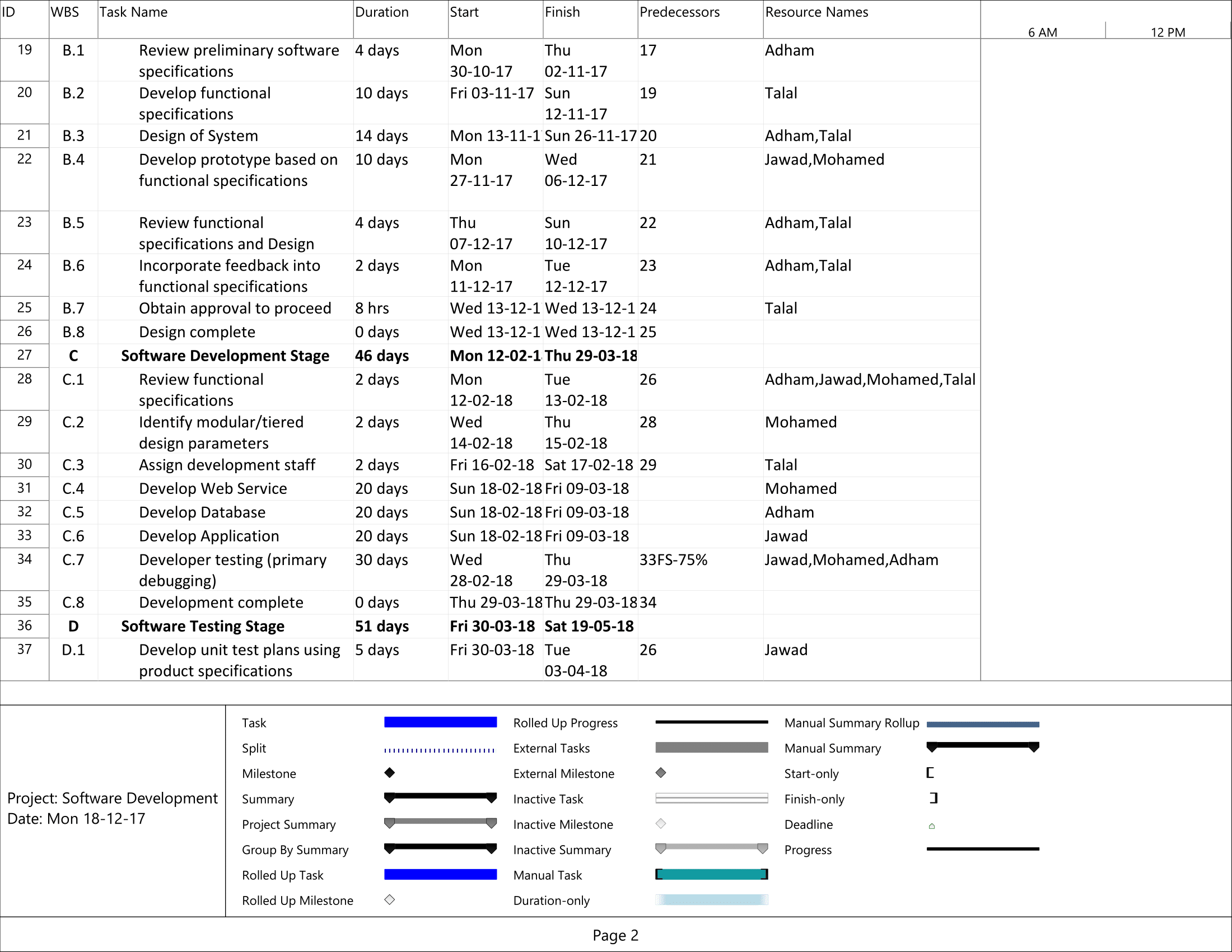
Fig 2. Gantt Chart (i)

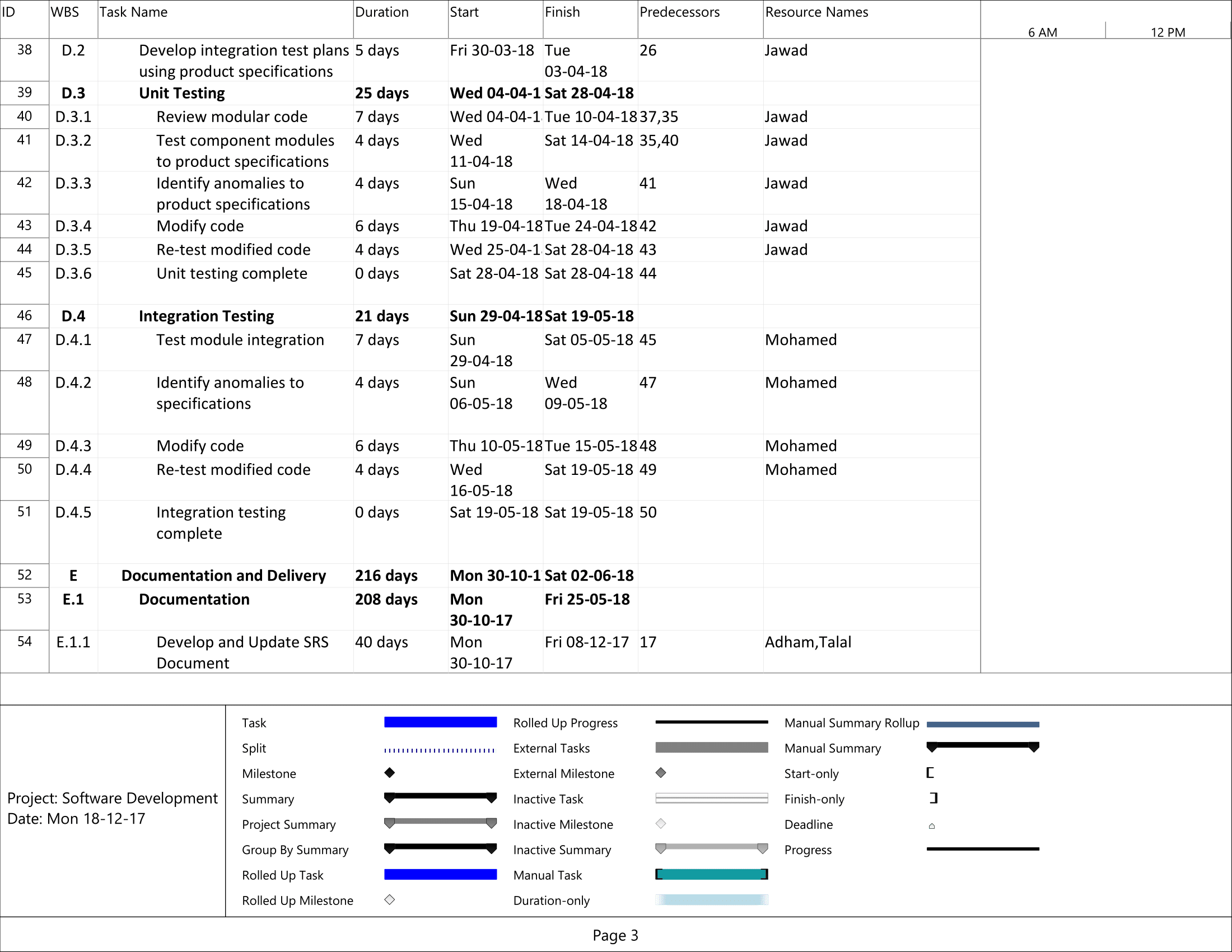
gantt dates

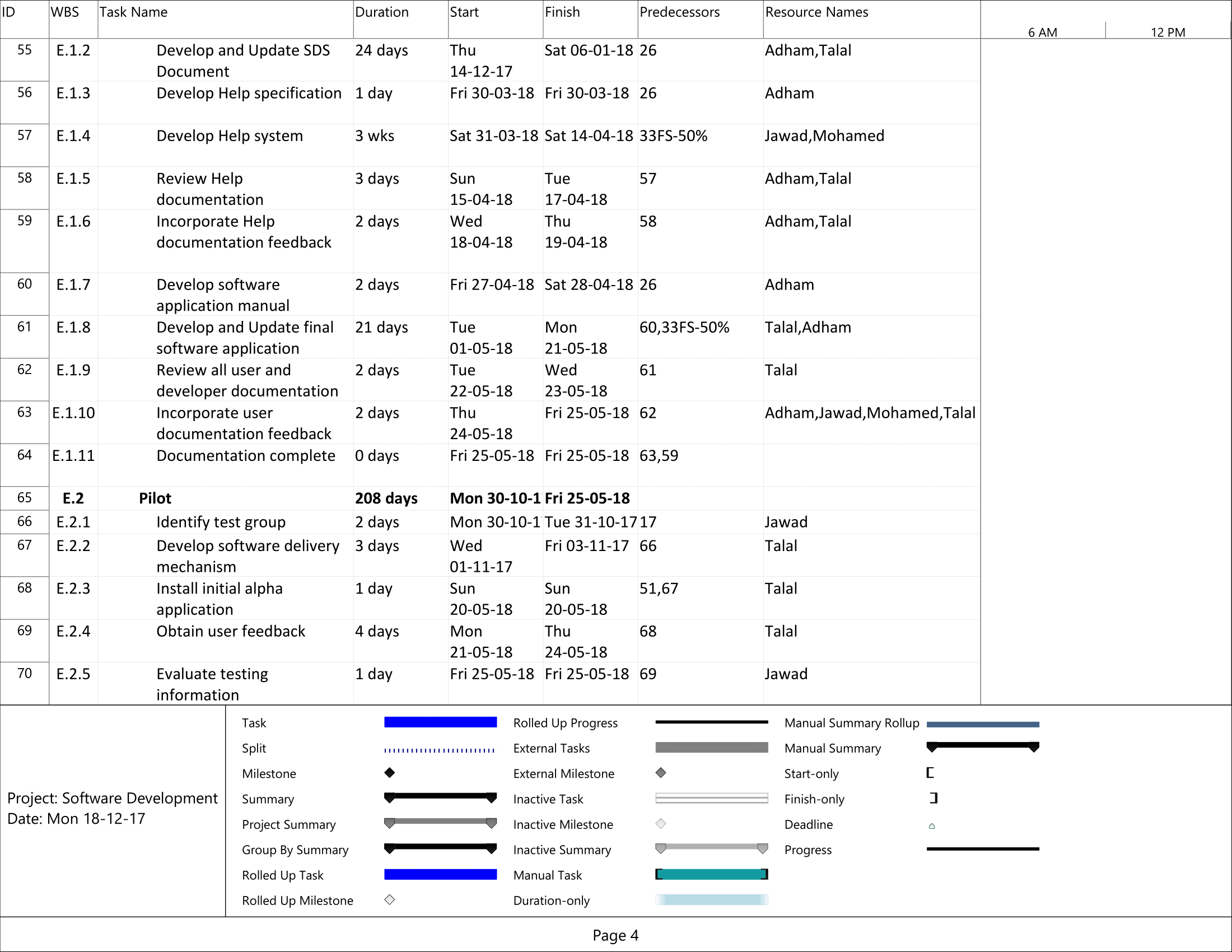
Description: C:\Users\Talal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\legend footer.png

Fig 3. Gantt Chart (ii)









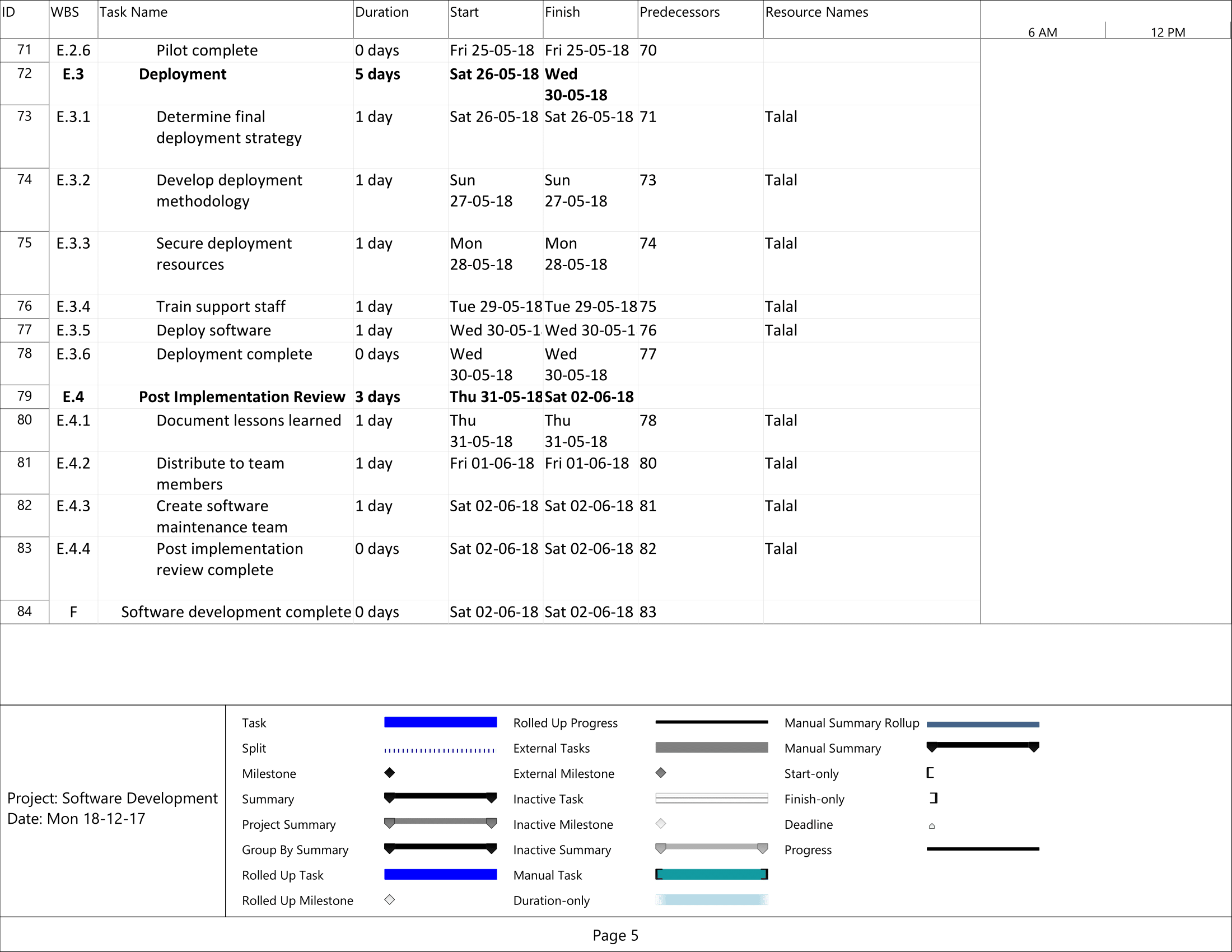
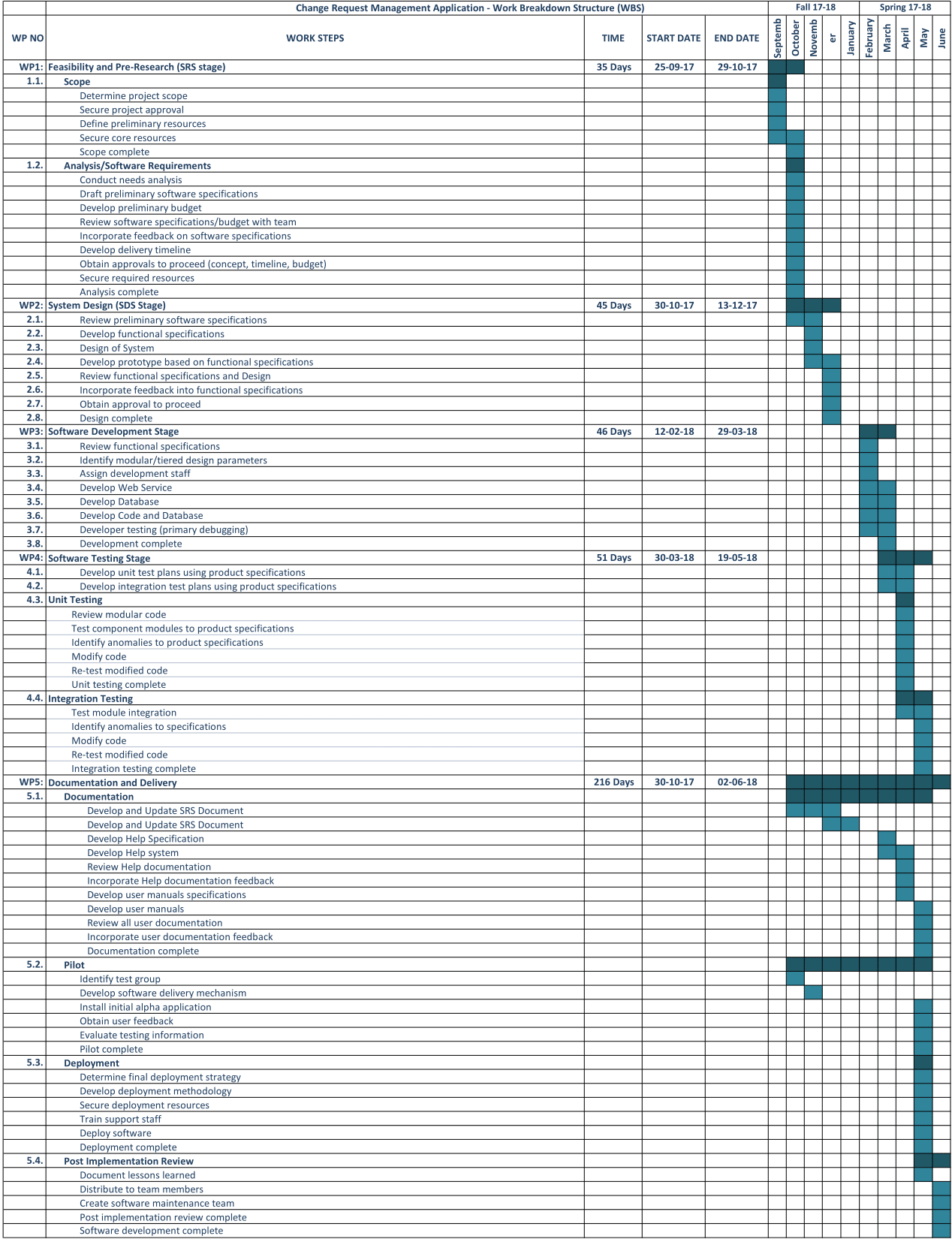


Table 8. WBS



## 2.9. List of Milestones

Table 9. List of Milestones

|  |  |  |
| --- | --- | --- |
| **No.** | **Description of Output** | **Expected Time Interval** |
| 1 | Scope determination and approval | |  |  | | --- | --- | | **25-09-17** | **01-10-17** | |
| 2 | Analysis/Software Requirements | |  |  | | --- | --- | | **02-10-17** | **29-10-17** | |
| 3 | System Design (SDS Stage) | |  |  | | --- | --- | | **30-10-17** | **13-12-17** | |
| 4 | Software Development Stage | |  |  | | --- | --- | | **12-02-18** | **29-03-18** | |
| 5 | Unit Testing | |  |  | | --- | --- | | **04-04-18** | **28-04-18** | |
| 6 | Integration Testing | |  |  | | --- | --- | | **29-04-18** | **19-05-18** | |
| 7 | Documentation | |  |  | | --- | --- | | **30-10-17** | **25-05-18** | |
| 8 | Pilot | |  |  | | --- | --- | | **30-10-17** | **25-05-18** | |
| 9 | Deployment | |  |  | | --- | --- | | **26-05-18** | **30-05-18** | |
| 10 | Post Implementation Review | |  |  | | --- | --- | | **31-05-18** | **02-06-18** | |
| 11 | Software Development complete | |  |  | | --- | --- | | **02-06-18** | **02-06-18** | |

Table 10. List of Risks

## 2.10. List of Risks

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Effects** | **Strategy** |
| The time required to develop the software is underestimated. | High | Serious | The most important requirements of the project should always be implemented first. We will have more time later on to implement the non-important requirements. |
| Software tools cannot work together in an integrated way. | High | Tolerable | Always minimize the number of design tools used and make sure that the outputs of these tools are compatible with each other. |
| Customers fail to understand the impact of requirements changes. | Moderate | Tolerable | Conduct frequent meetings with the stakeholders and keep being updated on latest requirement changes. |
| The rate of defect repair is underestimated. | Moderate | Tolerable | Replace potentially defective components with more reliable bought-in components. |
| The size of the software is underestimated. | Moderate | Insignificant | Investigate buying software components;  Investigate use of a program generator. |
| Code generated by code generation tools is inefficient. | Moderate | Insignificant | Risk is always expected since code generation tools often can’t produce reliable code . This code always needs editing by the software developers. |
| Key staffs are ill at critical times in the project. | Moderate | Serious | Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs. |
| The database used in the system cannot process as many transactions per second as expected. | Low | Serious | Investigate the possibility of buying a higher-performance database. |

## 2.11. Commercialization Potential

Commercialization of our product can start as soon as the development and testing of the most important modules is done. However, to further guarantee that everything is working as planned and to have an advantage over other applications, we are going to wait until most of the features of the application are done. After that, the commercialization process starts when the application is deployed to the Web Server of KKTCell. Then, many user feedbacks will be gained and improvements with new features and bug fixes will be implemented. In addition, during this time, an advertisement campaign in the KKTCell workplace could be made to promote our product and increase the user base. So the aim is to enable and invite many employees to use this application.

## 2.12. Project Economic Expectations

Table 11. Project Economic Expectations

|  |  |
| --- | --- |
| Time-to-market (month): | 9 |
| The expected increase in sales revenue (%): | 25% |
| The expected increase in market share (%): | 10% |
| Time to start to gain: | June 2018 |

## 2.13. Software Purchases

Table 12. Software Purchases

|  |  |
| --- | --- |
| **Change Request Management Application** | |
| **Line no** | **Instrument / Equipment / Software / Publication Name** | **No. of Item** | **Technical specification** | **Purpose of Project Activities** | **Post-Project Place of Use / Purpose** | | **Unit Price (USD)** | **Unit Price (TL)** | **Total Amount (TL)** |
| **R & D** | **Production** |
| 1 | Visual Studio | 4 | Integrated Development Environment (IDE) from Microsoft | Main IDE used for development of our project | No | Yes | 0 | 0 | 0 |
| 2 | Microsoft Project | 1 | Project Management Software | We will use this application to plan and schedule our project | No | Yes | 589.99 | 2600 | 2600 |
| 3 | Microsoft Office | 1 | An office suite of applications, servers, and services | Used in many areas of the project such as documentation | No | Yes | 399.99 | 1800 | 1800 |
| 4 | Modelio | 1 | Software Design Tool | Used to draw software design diagrams and generate code required for the application based on those diagrams | No | Yes | 0 | 0 | 0 |
| 5 | Bootstrap Studio | 1 | Software Design Tool and Code generator | Used to draw Web design diagrams and generate code required for the application based on those diagrams | No | Yes | 349 | 1570 | 1570 |
| 6 | Wireframe.cc | 1 | User Interface Design Tool | Used to draw a User Interface for our system | No | Yes | 99.99 | 450 | 450 |
| 8 | Oracle Database | 1 | Database for our Web Application | Permanent storage for all the user and application generated data | No | Yes | 350 | 1570 | 1570 |
| 9 | Slack | 4 | Cloud-based set of proprietary team collaboration tools and services. | Team Collaboration Tool for cooperating, sharing code and information with team members. | No | Yes | 0 | 0 | 0 |
| 10 | Visual Studio Team Services | 4 | Cloud service for collaborating on code development. | Version control tool similar to Git that allows the team to work together on the same code and project. | No | Yes | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  | TOTAL | 7990 TL |

## 2.14. Quarterly Estimated Cost Form (TL)

Table 13. Quarterly Estimated Cost Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Change Request Management Application** | | | | |
| **Cost Item** | **2017-2018** | | **TOTAL**  **(TL)** | **TOTAL COST RATE OF CONTENTS (%)** |
| **I** | **II** |
| **Personnel** | 20000 | 20000 | 40000 | 58.83 |
| **Travel** | 1000 | 1000 | 2000 | 2.94 |
| **Software Costs** | 7990 | - | 9340 | 13.74 |
| **Research, Development and Testing Costs** | 3100 | 3100 | 6200 | 9.12 |
| **Domestic Deployment Services and Maintenance Procurement Cost** | 1800 | 1800 | 3600 | 5.29 |
| **Overseas Deployment Services and Maintenance Procurement Cost** | 2000 | 2000 | 4000 | 5.88 |
| **Research Material** | 2100 | 2100 | 4200 | 6.17 |
| **TOTAL COST** | **37,990 TL** | **30,000 TL** | **67,990 TL** | **On a scale of 100** |
| **IN THE PROJECT TOTAL MAN-MONTH** | | | **720 hours** | |

# 3. REQUIREMENTS ANALYSIS

## 3.1. Functional Requirements

Login and Logout:

**Requirement 1**

For one user the login-state can be either logged in or logged out.

**Requirement 2**

The system shall store the login-state in a server session.

**Scenario 1: User login**

Precondition: The user is not logged in.

1. The user accesses the system.

2. The user is asked to provide username and password on a log-in page.

3. The user provides a correct username and password.

4. The user is logged in and a functionality page is shown

**Requirement 3:**

Scenario 1 should be supported by the system.

**Scenario 2: User logout**

Precondition: The user is logged in.

1. The user accesses the system.

2. The user is presented to a page that includes a logout link.

3. The user requests to be logged out.

4. The user is logged out and informed about this through on the next page that is displayed.

**Requirement 4**

Scenario 2 should be supported by the system.

**Scenario 3: Failed user login**

Precondition: The user is not logged in.

1. The user accesses the system.

2. The user is asked to provide a user name and password on a log-in page.

3. The user provides a user name and password that is not registered in the database.

4. The user is not logged in and an error message is shown. The user is again asked to provide user name and password

**Requirement 5**

Scenario 3 should be supported by the system.

**Requirement 6**

When a user reaches the system and is not logged in he/she should be asked to provide a username and a password. No other information should be provided to the user.

**Requirement 7**

When a user submits a username and a password they should be compared to the list of users and if the user should be granted access to the system the server-state should be changed to “logged in” and the functionality page shown.

**Requirement 8**

All pages shown to a logged in user should include a log out functionality, e.g. a button for logging out of the system.

**Requirement 9**

If a logged in user is inactive for longer than 20 minutes he/she should be logged out and required to log in again before continuing using the system.

**Requirement 10**

The system should be able to handle and detect incorrect input. No type of incorrect input should be able to crash the system or corrupt the data in the system.

Homepage:

**Requirement 11**

Each user will be shown a different homepage based on his/her title and department.

**Requirement 12**

Each user will be shown different content on their homepage.

**Requirement 13**

All homepages show user details:

1. Employee name, surname.
2. Employee number
3. Employee title.
4. Employee’s manager/leader.
5. Employee enrolled department/team.

**Requirement 14**

All homepages show statistics:

1. Number of active requests.
2. Number of closed requests.
3. Notifications

**Requirement 15**

“Number of active requests” is a clickable hyperlink that will show active requests.

**Requirement 16**

“Number of closed requests” is a clickable hyperlink that will show closed requests.

**Requirement 17**

All homepages have a side menu on the left, that includes:

1. Create new request.
2. View/update active requests.
3. View requests history.
4. Change password.

**Requirement 18**

“Create new request” is a clickable button.

**Requirement 19**

“View/update active requests” is a clickable button that will view active requests and prompt for update or cancelation.

**Requirement 20**

“View requests history” is a clickable button that will show previously created and closed requests.

**Requirement 21**

“Change password button” is a clickable button that will show the password change form.

**Request Creation:**

**Scenario 4**

When a user clicks on “Create new request”:

1. Side menu will be automatically hidden.
2. The “Request Creation Form” will be shown.

**Requirement 22**

Scenario 4 should be supported by the system.

**Requirement 23**

“Request Creation Form” consists of 0-7 panels.

**Requirement 24**

“Panel 0” Shows general information of the Functional Requirement Document (FRD).

**Requirement 25**

General information is stored in “Panel 0” contains:

1. Request No. or FRD No.
2. Request Name.
3. Request Creator
4. Request version number.

**Requirement 26**

“Panel 0” contains an empty table of all versions to be submitted for the new request.

**Requirement 27**

“Panel 1” shows list of requirements to be added for the FRD.

**Requirement 28**

Every requirement (item) in “Panel 1” will contain:

1. Version number.
2. Username of the creator.
3. Date created.

**Requirement 29**

“Panel 2” shows a table list of target audience that can be chosen individually.

A minimum of one selection is required.

**Requirement 30**

List of target audience in “panel 2” is updated and retrieved automatically from the applicable table in the database.

**Requirement 31**

“Panel 3” shows a table that contains a list of channels that can be affected by the demand.

A minimum of one selection is required.

**Requirement 32**

List of channels that can be affected by the demand in “panel 3” is updated and retrieved automatically from the applicable table in the database.

**Requirement 33**

“Panel 4” shows requests on SMS content, through two types of SMS requests:

1. “New Campaign Request”
2. “Old Campaign Request\*”

\*Not shown when a new request is created.

**Requirement 34**

“Requests on SMS content” will be shown in a table in “panel 4” containing the following data:

1. Sender.
2. Code.
3. Explanation.
4. Content (TR).
5. Length.
6. Content (ENG).
7. Length.
8. Username.
9. Date.

**Requirement 35**

“Code” is assigned/retrieved from the applicable table in the database.

**Requirement 36**

“Explanation” is assigned/retrieved from the applicable table in the database.

**Requirement 37**

“Content (TR)” is a textbox that can be filled within the current page, Turkish characters must be checked and not allowed.

**Requirement 38**

“Content (ENG)” is a textbox that can be filled within the current page.

**Requirement 39**

“Length” contains a function that counts the number of characters in and maximum 254 characters are allowed.

**Requirement 40**

“Username” should be updated and assigned automatically with the name of the user responsible for the creation/update taking place.

**Requirement 41**

“Date” should be updated and assigned automatically with the current date.

**Requirement 42**

“Panel 5” contains an attachment form that accepts files of the following types:

1. Pictures (JPEG, PNG, JPG)
2. Videos (MP4, AVI, MKV, MPEG-4)
3. Documents (DOCX, PDF, XLS

**Requirement 43**

“Panel 6” contains a table of current packages and tariffs, listed by priority.

**Requirement 44**

Employee can change the discount of each package.

**Requirement 45**

The table of current packages will be automatically rearranged according to the changes in discounts made by employees.

**Requirement 46**

“Panel 7” contains list of required confirmations from applicable departments and employees.

**View/Update active requests:**

**Scenario 5**

When a user clicks on “View/Update request”:

1. Side menu will be automatically hidden.
2. List of active requests related/assigned to the logged in user will be shown.

**Requirement 47**

Scenario 5 should be supported by the system.

**Requirement 48**

“Panel 0” Shows general information of the Functional Requirement Document (FRD).

**Requirement 49**

General information stored in “Panel 0” contains:

1. Request No. or FRD No.
2. Request Name.
3. Request Creator
4. Request version number.

**Requirement 50**

“Panel 0” contains a table of all versions submitted for the selected request.

Update can take place.

**Requirement 51**

“Panel 1” shows list of requirements added to the FRD.

Update can take place.

**Requirement 52**

Every requirement (item) in “Panel 1” contains:

1. Version number.
2. Username of the creator.
3. Date created.

**Requirement 53**

“Panel 2” shows a table list of target audience that can be chosen individually.

A minimum of one selection is required.

**Requirement 54**

List of target audience in “panel 2” is updated and retrieved automatically from the applicable table in the database.

**Requirement 55**

“Panel 3” shows a table that contains a list of channels that can be affected by the demand.

A minimum of one selection is required.

**Requirement 56**

List of channels that can be affected by the demand in “panel 3” is updated and retrieved automatically from the applicable table in the database.

**Requirement 57**

“Panel 4” shows requests on SMS content, through two types of SMS requests:

1. “New Campaign Request”
2. “Old Campaign Request”

**Requirement 58**

“Requests on SMS content” will be shown in a table in “panel 4” containing all submissions and data entries made before with the following data:

1. Sender.
2. Code.
3. Explanation.
4. Content (TR).
5. Length.
6. Content (ENG).
7. Length.
8. Username.
9. Date.

**Requirement 59**

“Code” is assigned/retrieved from the applicable table in the database.

**Requirement 60**

“Explanation” is assigned/retrieved from the applicable table in the database.

**Requirement 61**

“Content (TR)” is a textbox that can be filled within the current page, Turkish characters must be checked and not allowed.

**Requirement 62**

“Content (ENG)” is a textbox that can be filled within the current page.

**Requirement 63**

“Length” contains a function that counts the number of characters in and maximum 254 characters are allowed.

**Requirement 64**

“Username” should be updated and assigned automatically with the name of the user responsible for the creation/update taking place.

**Requirement 65**

“Date” should be updated and assigned automatically with the current date.

**Requirement 66**

“Panel 5” contains an attachment form that accepts files of the following types:

1. Pictures (JPEG, PNG, JPG)
2. Videos (MP4, AVI, MKV, MPEG-4)
3. Documents (DOCX, PDF, XLS

**Requirement 67**

“Panel 6” contains a table of current packages and tariffs, listed by priority.

**Requirement 68**

Employee can change the discount of each package.

**Requirement 69**

The table of current packages will be automatically rearranged according to the changes in discounts made by employees.

**Requirement 70**

“Panel 7” contains list of required confirmations from applicable departments and employees.

**Requirement 71**

If a user requests a new confirmation to be made on an active request, a notification should be sent to the applicable request creator.

**View requests history:**

**Requirement 72**

Users can view a list of previously created and closed (inactive) requests.

No updates can take place.

**Requirement 73**

When a user clicks on a closed request from the list, the request panels will be shown and the side menu will be hidden. No updates on the panels and their information can take place.

**Change password:**

**Requirement 74**

“Change Password” button will show a password change form that contains:

1. Old password field
2. New password field
3. Retype new password field

**Requirement 75**

Old password entered is compared and authenticated with the password currently stored in the database. New password should match the password in the “Retype new password field” and it should be compatible with the security requirements set.

## 3.2. Non-Functional Requirements

### 3.2.1. Software Quality Attributes

Adaptability, availability, correctness, flexibility, maintainability, portability, reliability, reusability, robustness, testability, and usability.

### 3.2.2. Performance Requirements

The response time after each request to the web server must not exceed 3 seconds.

### 3.2.3. Safety Requirements

The application should not contain any safety threats, malware or adware to users.

### 3.2.4. Implementation Requirements

The Database should be implemented using an Oracle and the project must be an MVC based project with C# as a server side language.

### 3.2.5. Security Requirements

The system shall be a very secure system and it should implement the SHA-2 Hashing Standard as set out by the U.S. National Security Agency (NSA) in 2001 to secure the passwords within the database of the application. Only the people authorized to access the system shall be able to access it.

* SHA-256 Hashing with Salt Encryption.
* HTTPS Secured Web Application.
* Session key with a 20 minute length.
* Secure JavaScript coding practices.

## 3.3 Ethical issues

The user of this system may write some sensitive personal information in the forms of the application. Another unauthorized user may access this sensitive information. That’s why the application should make sure that the privacy law standards are set where the application is to be distributed.

# 4. DESIGN

## 4.1. High level design

### 4.1.1. System Architecture - MVC

We implemented an MVC architectural design pattern for our Web Application, as depicted in Figure 4. The MVC architecture separates an application into three main components: the model, the view, and the controller.

* Models: Model objects are the parts of the application that implement the logic for the application's data domain. Model objects retrieve and store model state in a database.
* Views: Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data.
* Controllers: Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI.

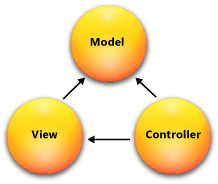


Fig 4. MVC Architecture

### 4.1.2. Network Architecture

We will implement a Client Server Architecture for the Network of our system. The Web Application will be installed on all PC terminals at the KKTCell Workplace and all the terminals will be connected to a single Web Server. This network model is depicted in Figure 5.

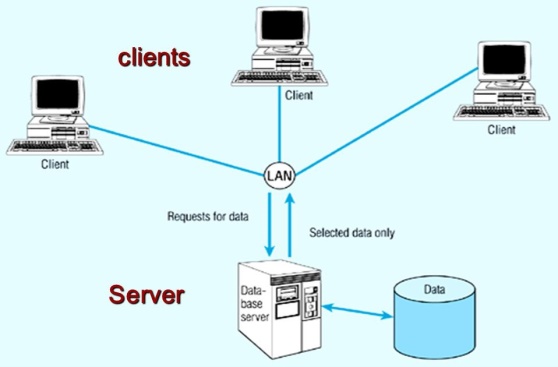


Fig 5. Client-Server Architecture

### 4.1.3. Context Diagram

A context diagram is considered the highest level of design in Dataflow diagrams. The Context diagram of Change Request Management Application was created using the draw.io tool and it is shown in Figure 6.

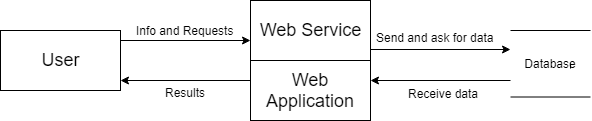
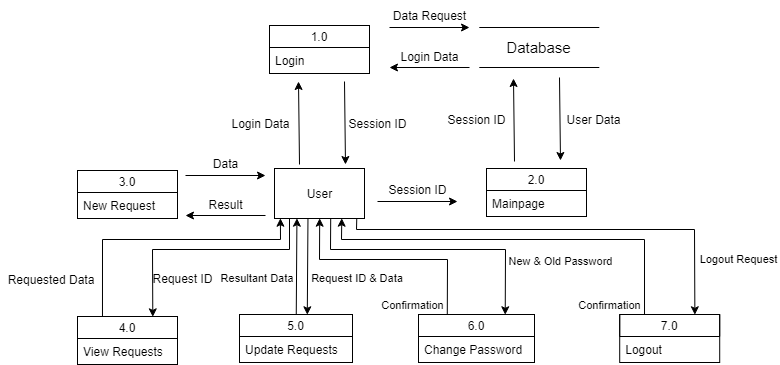


Fig 6. Context Diagram

### 4.1.4. Dataflow Diagram – Level 0

The second highest level in Dataflow diagram designs is a Level 0 diagram. The Level 0 diagram for Change Request Management Application which was created using the draw.io tool is shown in Figure 7.

  
 Fig 7. Level 0 Data Flow Diagram

### 4.1.5. User Interface Design

Each user interface for Change Request Management Application will be explained along with some screenshots in this Section. These user interfaces were created using the Wireframe.cc tool. These user interfaces may or may not have the same look as the final product.

**4.1.5.1. Login Page Design**

When the user first runs the application, this page will be shown. The user must login to access the system. The SHA-256 with salt hashing Standard will be implemented to secure the password of the user. This is shown in Figure 8.

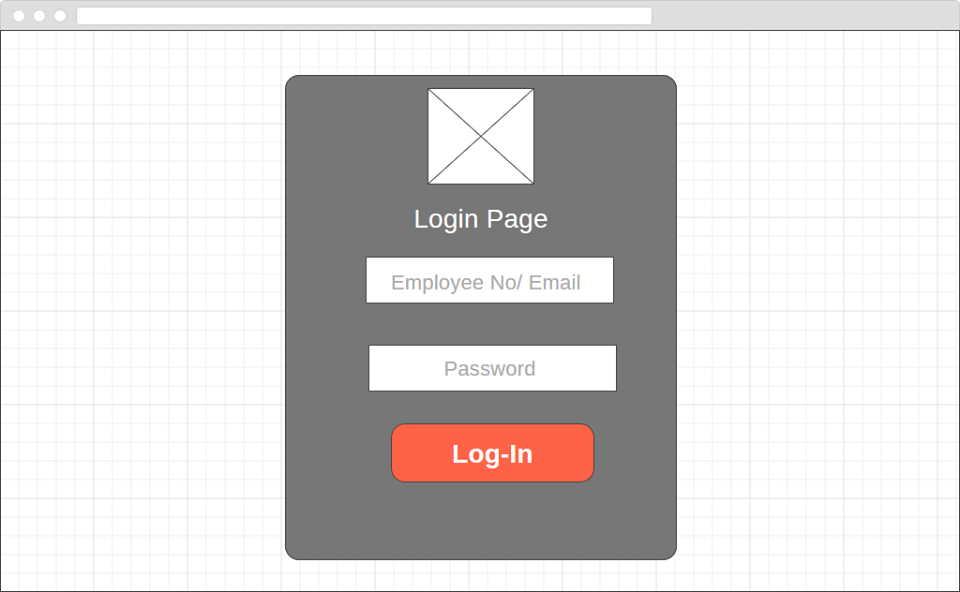


Fig 8. Login Page Design

**4.1.5.2. Home Page Design**

A depiction of the default page shown directly after logging in is shown in figure 9.

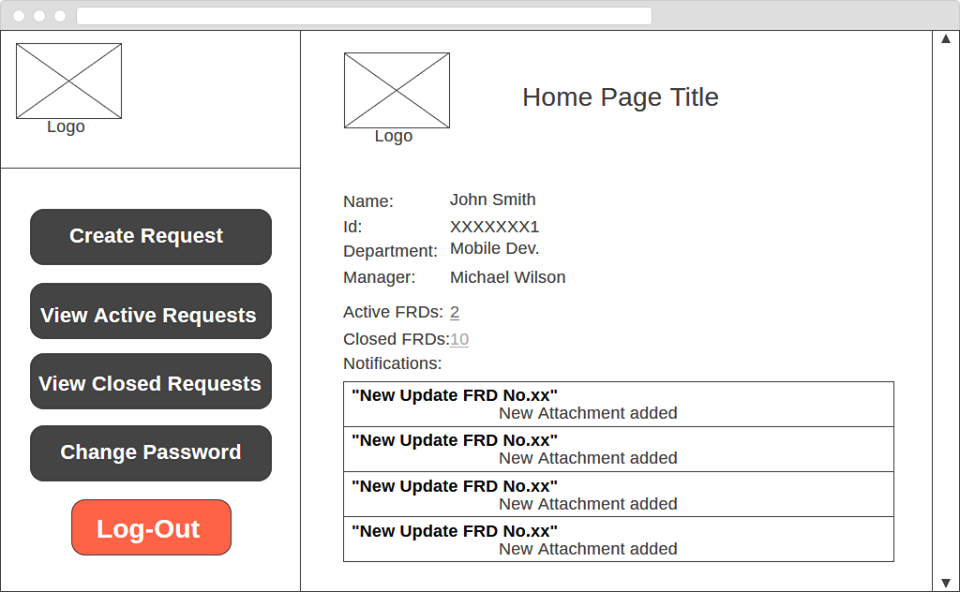


Fig 9. Home Page Design

**4.1.5.3 New Request Page Design**

The page shown after choosing to create a new request is shown in figure 10.

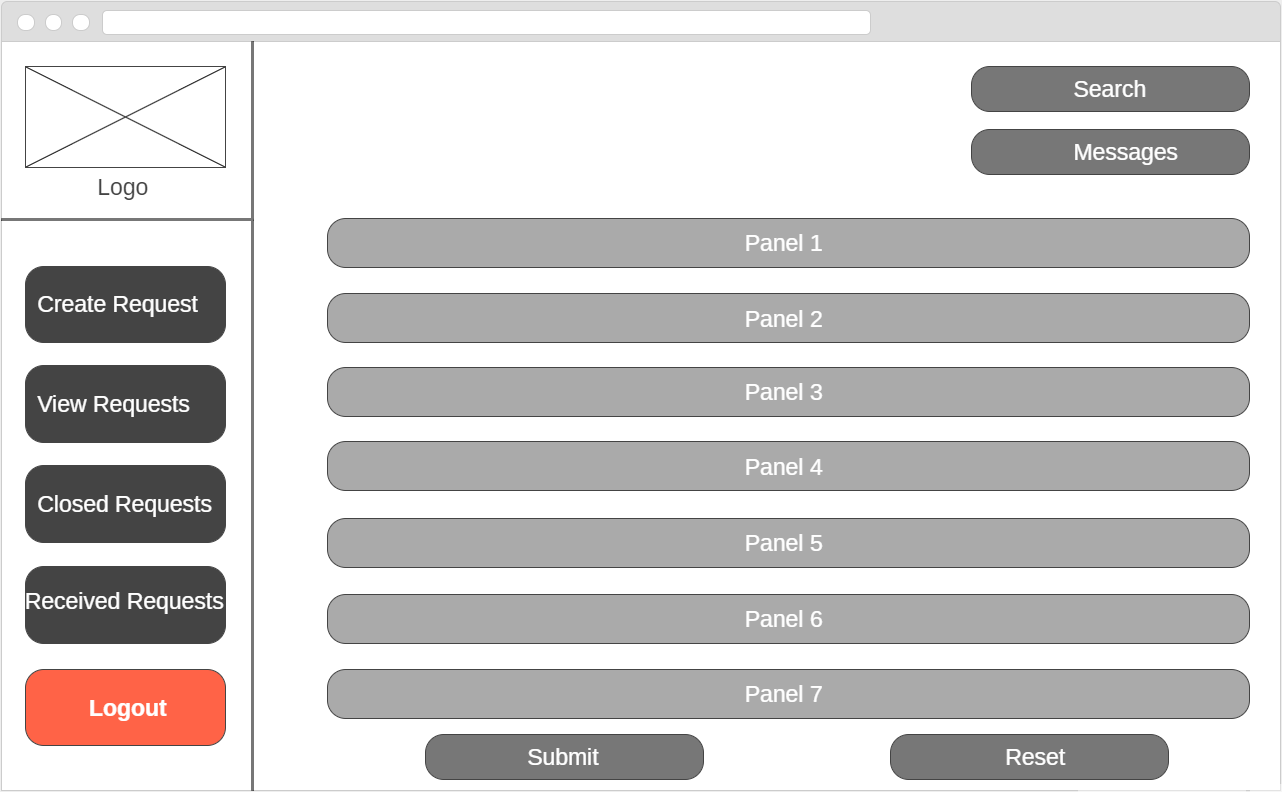


Fig 10. New Request Page Design

**4.1.5.4. Panel 1 Design – General Information**A depiction of the proposed design of panel 1 of the Request form is shown in figure 11.

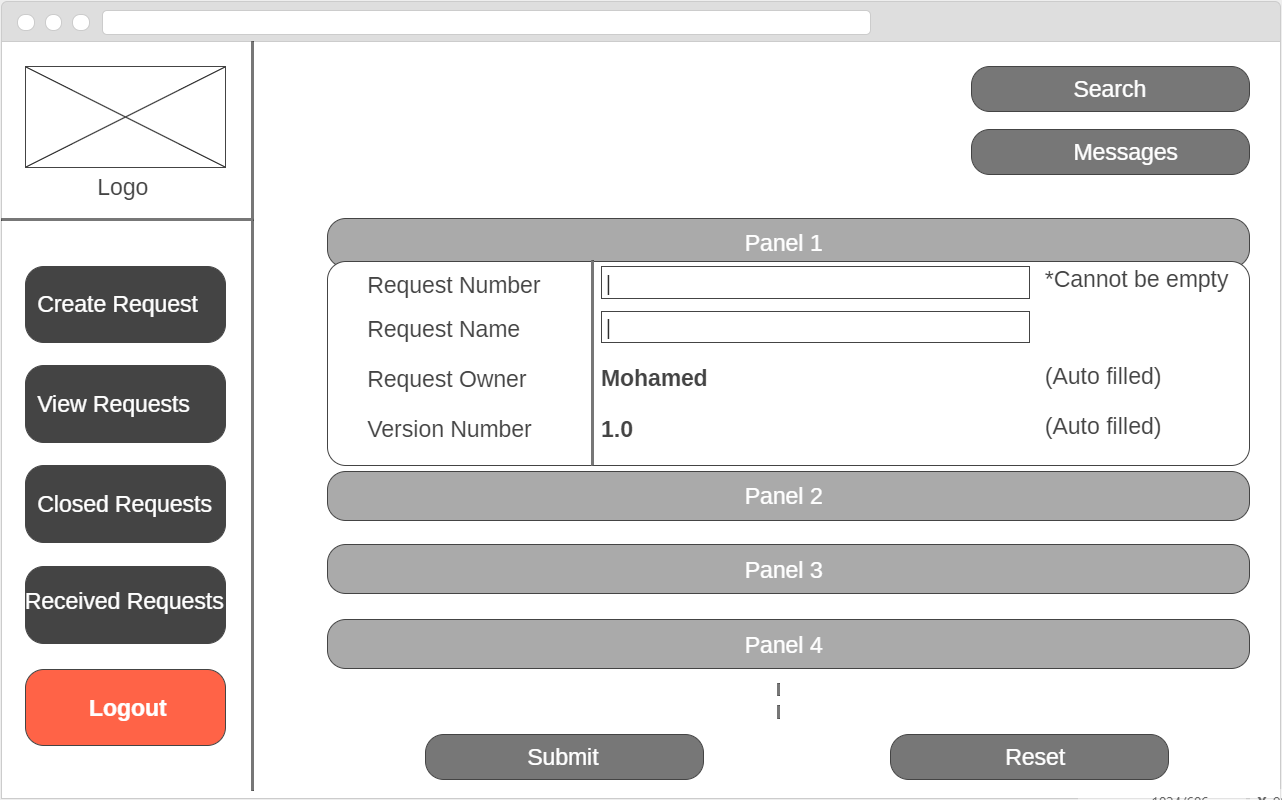


Fig 11. Panel 1 Design

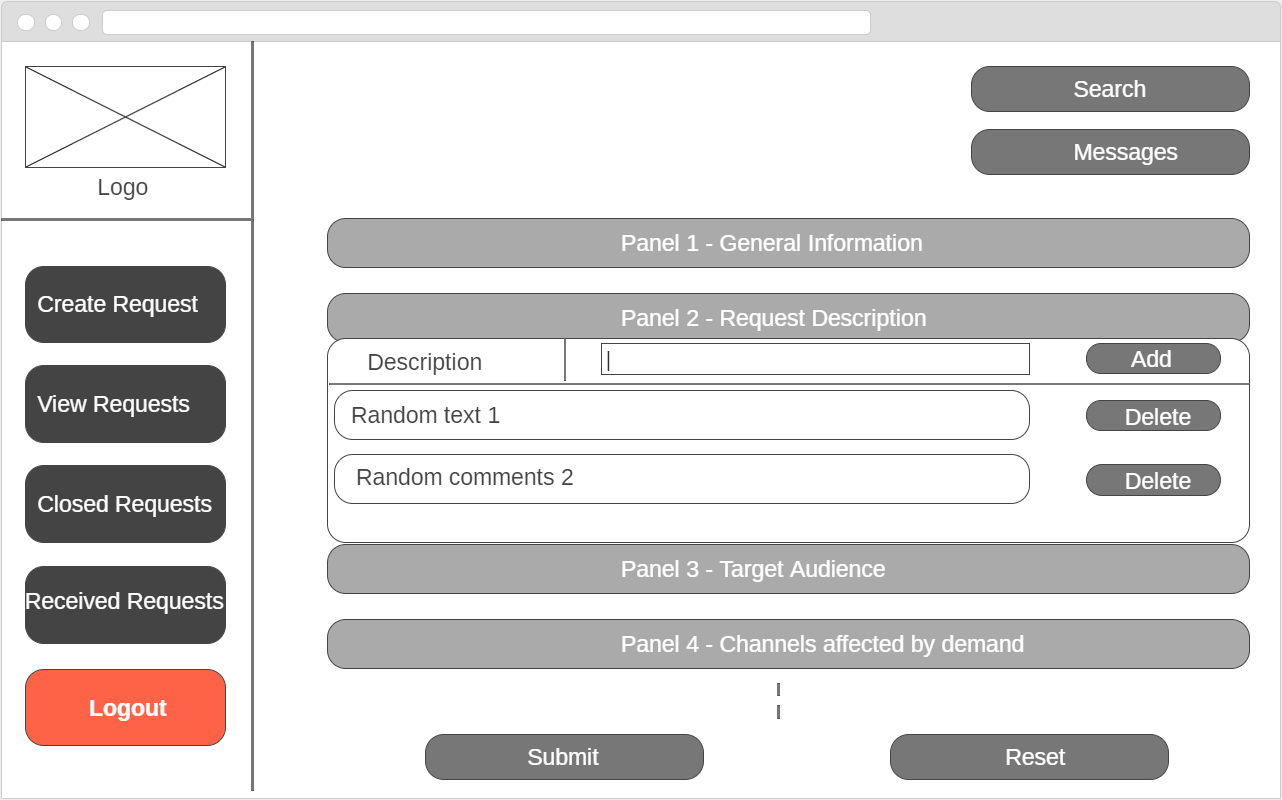
**4.1.5.5. Panel 2 Design – Request Description**  
A depiction of the proposed design of panel 2 of the Request form is shown in figure 12.

Fig 12. Panel 2 Design

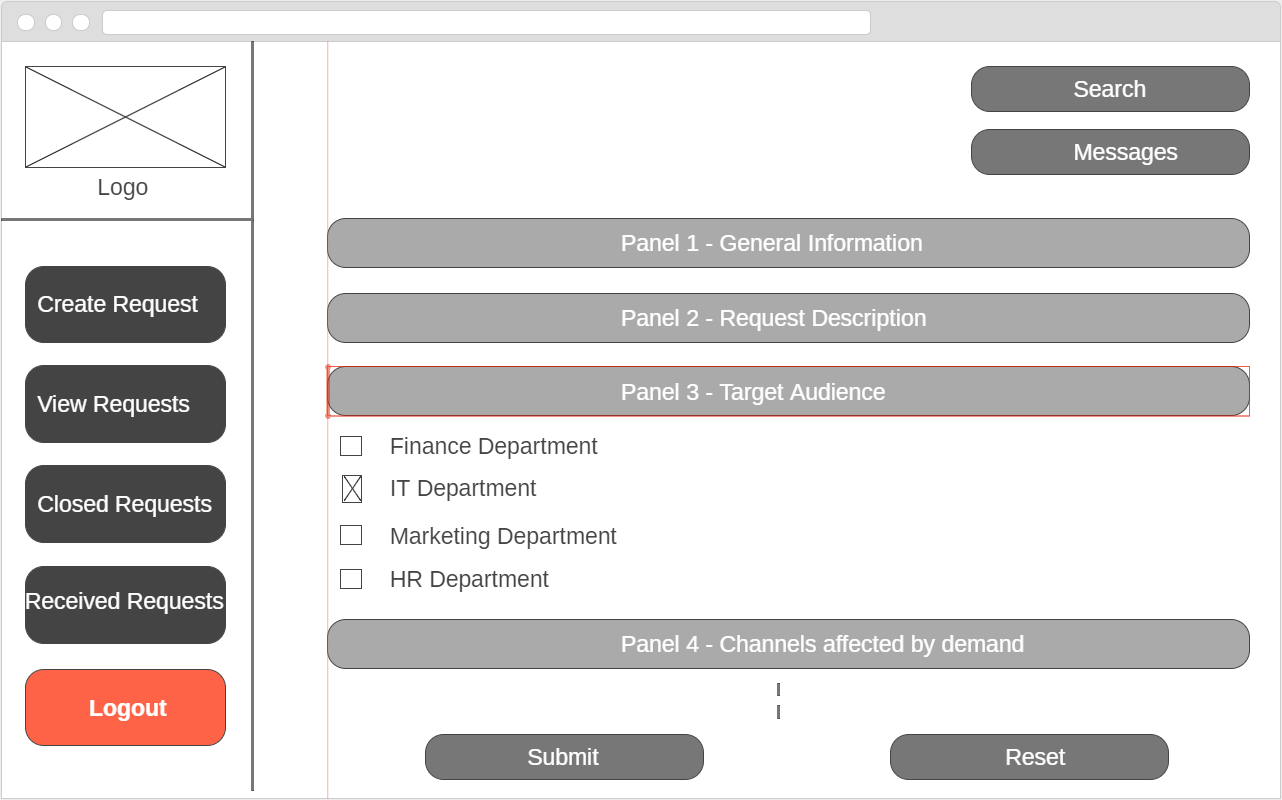
**4.1.5.6. Panel 3 Design – Target Audience**  
A depiction of the proposed design of panel 3 of the Request form is shown in figure 13.

Fig 13. Panel 3 Design

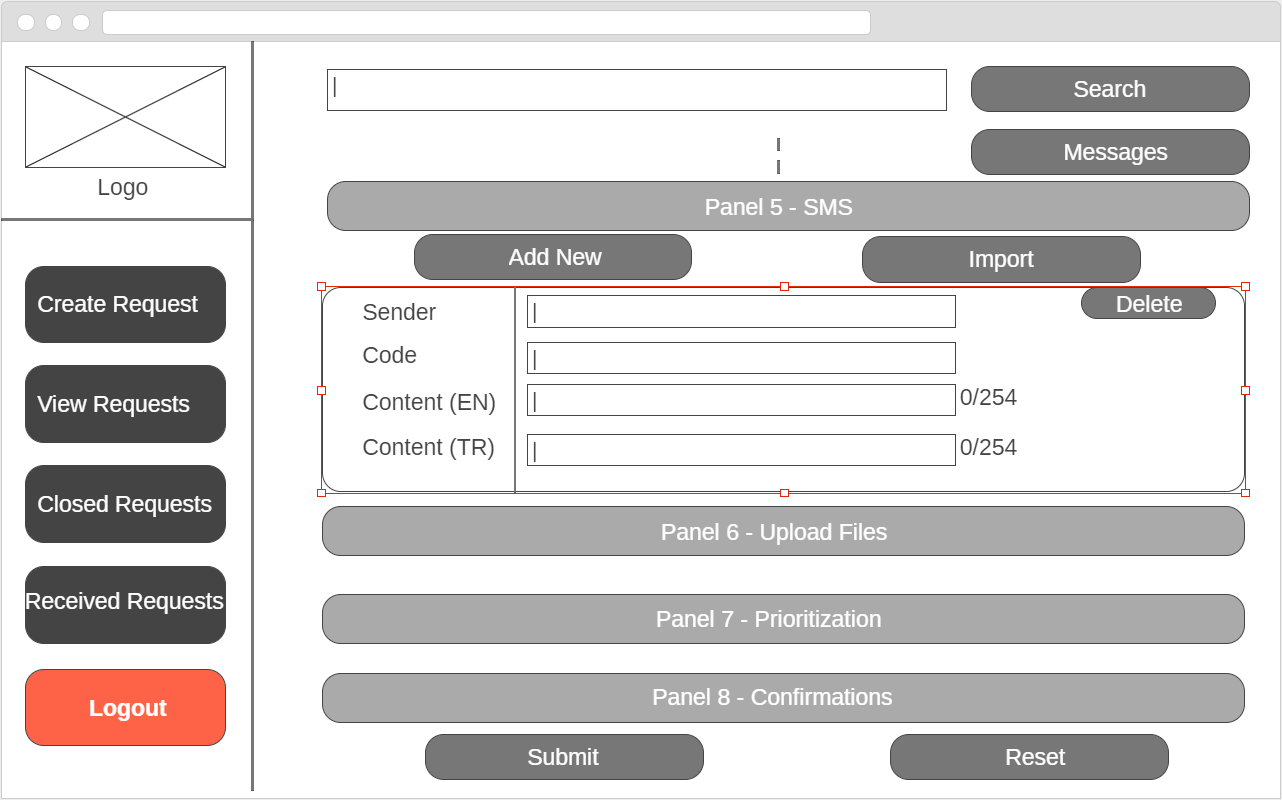
**4.1.5.7. Panel 5 Design – Add New SMS**  
A depiction of the proposed design of panel 5 (Add new SMS) of the Request form is shown in figure 14.

Fig 14. Panel 5 Design – Add New SMS

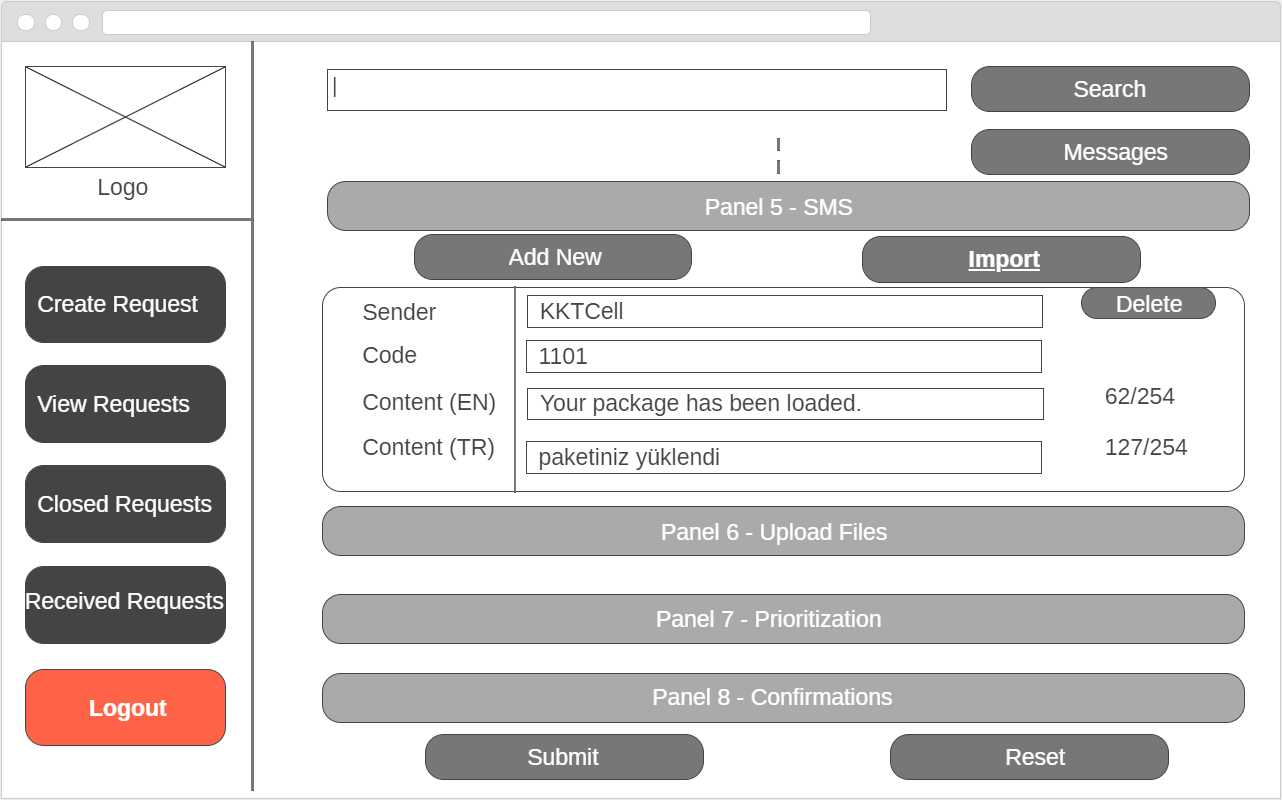
**4.1.5.8. Panel 5 Design – Import SMS**  
A depiction of the proposed design of panel 5 (Import SMS) of the Request form is shown in figure 15.

Fig 15. Panel 5 Design – Import SMS

**4.1.5.9. Panel 8 Design - Confirmations**  
A depiction of the proposed design of panel 8 of the Request form is shown in figure 16.

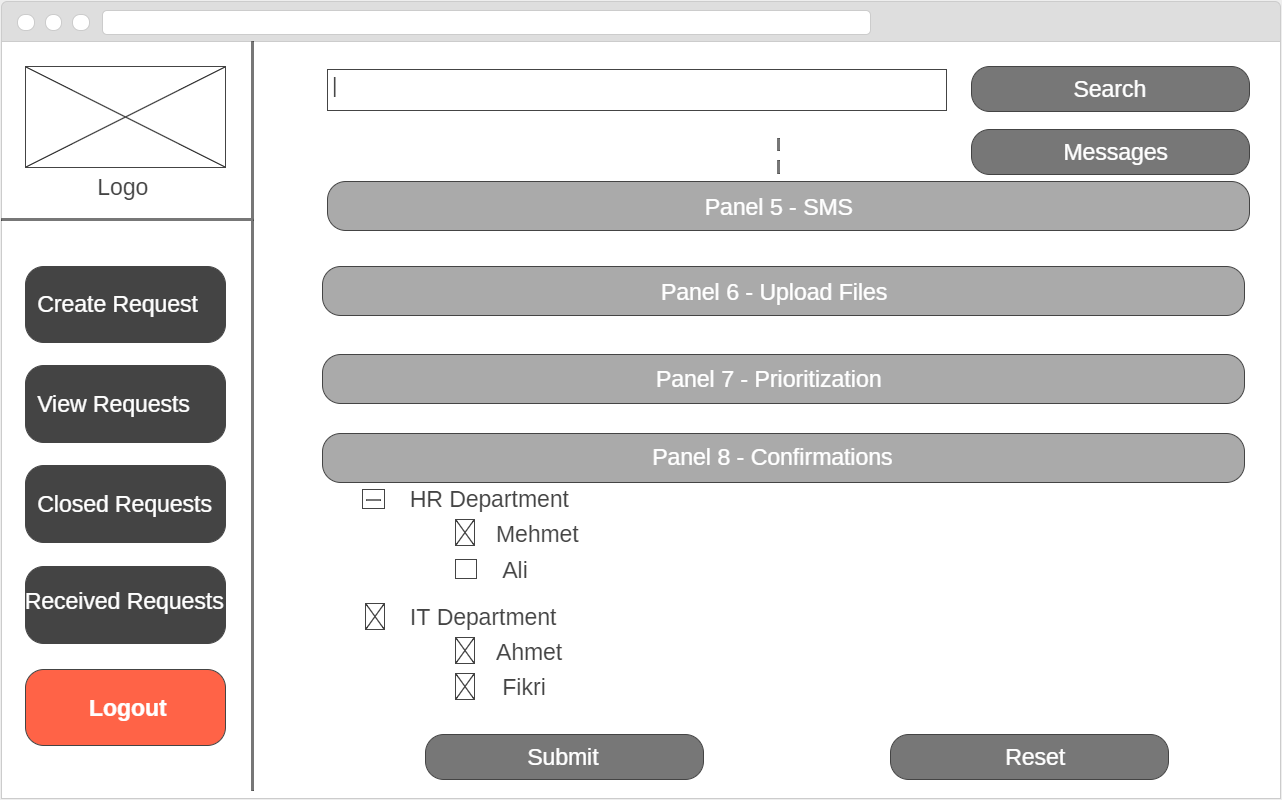


Fig 16. Panel 8 Design

## 4.2. Low level design

### 4.2.1. Dataflow Diagram – Level 1

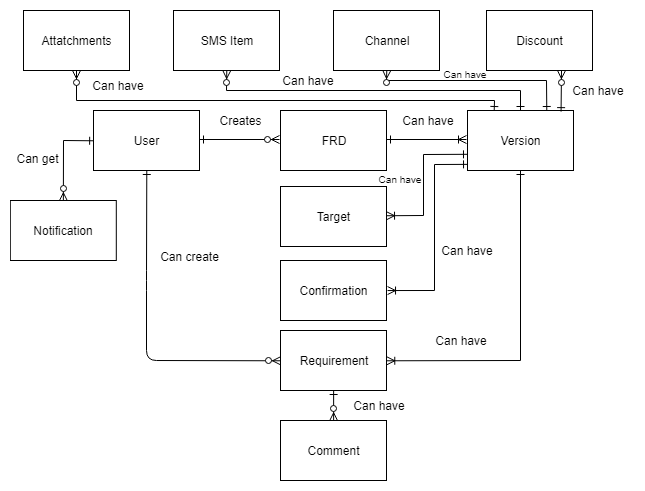
This level 1 dataflow diagram in figure 17 describes how data flows in the important Report Form Updating Module of the Change Request Mangment System. This diagram was created using the draw.io tool.



Fig 17. Level 1 Data Flow Diagram (Form Updating)

### 4.2.2. Data Dependencies

This will be represented as an Entity Relationship Diagram in Figure 18. This diagram was created using the draw.io tool.

  
 Fig 18. Entity Relation Diagram

### 4.2.3. Class Diagram

The class diagram in figure 19 shows the main classes of Change Request Management Application and the relations between them. The Class diagram was created using the draw.io tool.

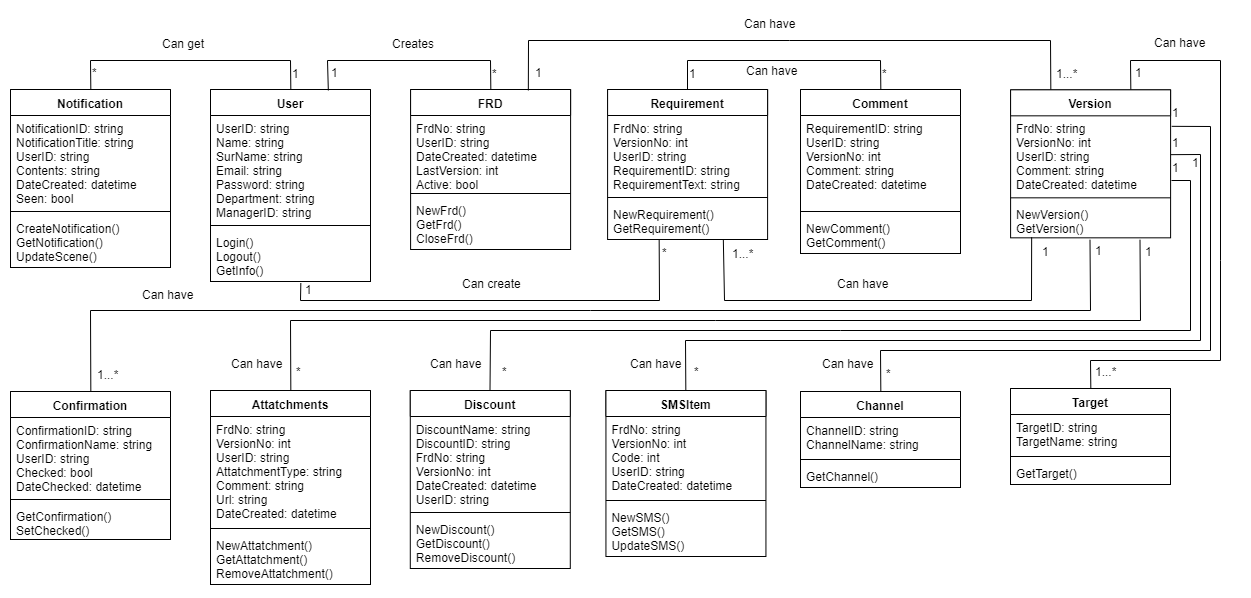
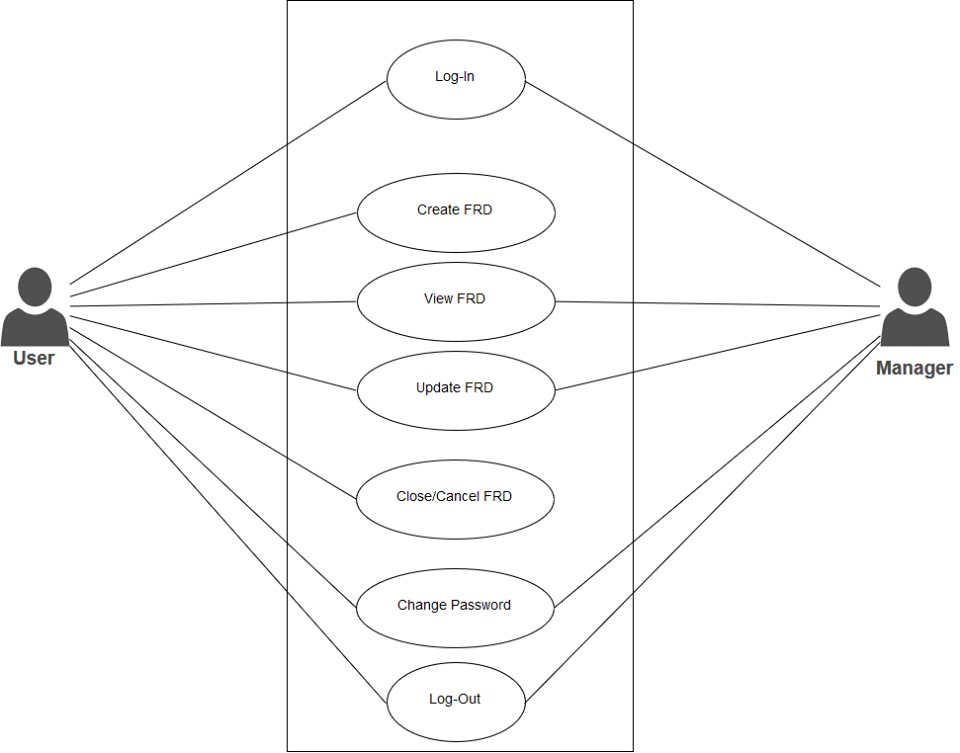
****

Fig 19. Class Diagram

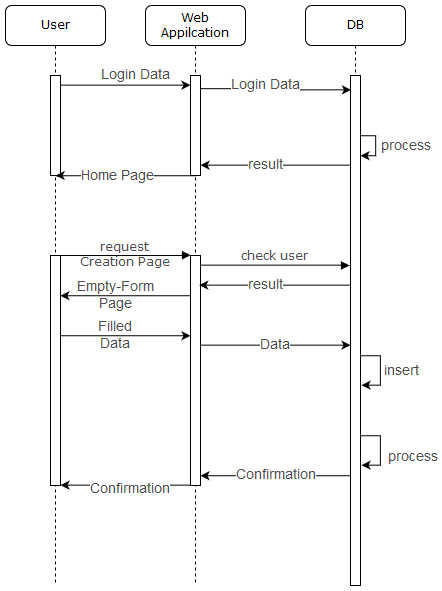
### 4.2.4. UML Interaction Diagrams

A use case diagram which was created using the Modelio tool is shown in figure 20. The sequence diagram shown in Figures 21 was created using the Modelio tool. An activity diagram is shown in Figure 22 which was created using draw.io tool. A business process Model which was created using the Modelio tool is shown in figure 23.

4.2.4.1. Use Case Diagram

  
 Fig 20. Use Case Diagram

4.2.4.2. Sequence Diagram – Login to System

  
 Fig 21. Sequence Diagram – Login to System

4.2.4.3. Activity Diagram – View Request

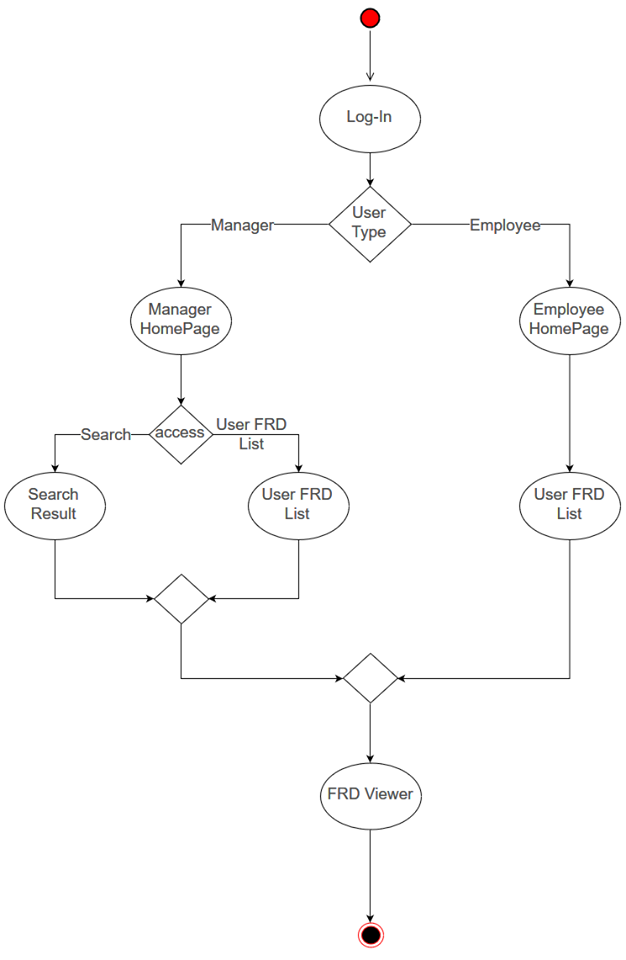


Fig 22. Activity Diagram – View Request

4.2.4.4. Business Process Model Notation

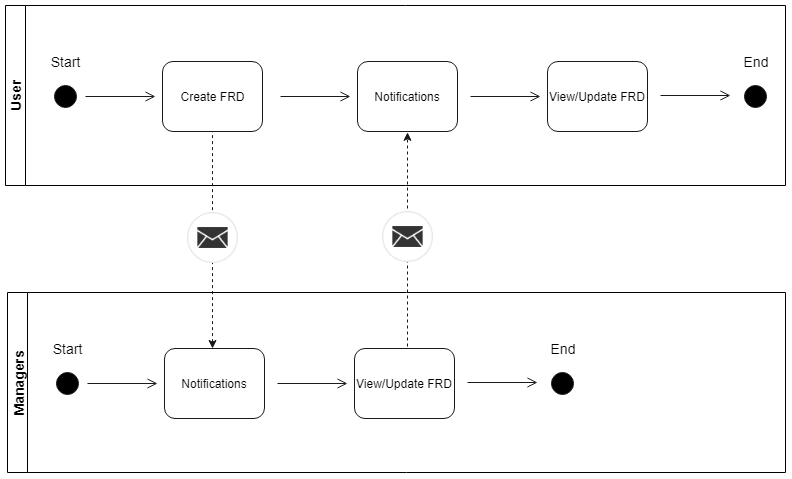


Fig 23. Business Process Model

# 5. IMPLEMENTATION

## 5.1. Tools, technologies, platforms, and libraries used

|  |  |  |
| --- | --- | --- |
| HTML | Visual Studio IDE | JQuery |
| CSS | ASP.NET RESTful Web API | React |
| JavaScript | JSON | LINQ |
| C# Programming Language | Bootstrap | Webpack |
| Oracle Database | Sass | Babel |
| ASP.NET MVC | Google Chrome Developer Tools | List.js |
| Slack | Visual Studio Team Services |  |

## 5.2. Use of Software Engineering Process Steps

These steps were carried out in order to build the project:

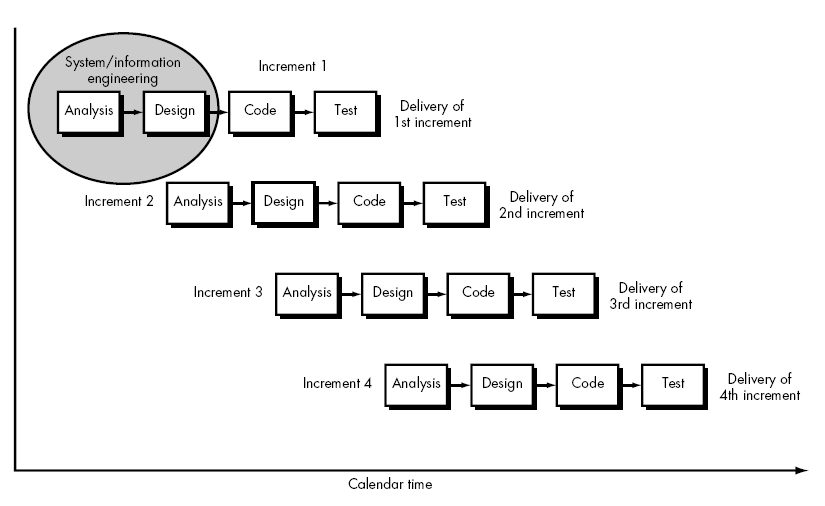
1. Determination of an ideal Software Development Approach: Incremental Model. This was discussed in Section 2.6 of the report and is shown in figure 24.

Fig 24. Incremental Method of Development

2. Make a Software Development Plan with the help of Microsoft Project tool.

3. Arrange the features of the system according to their importance.  
 High Priority:  
 a. Login  
 b. Create new FRD  
 c. View FRD  
 d. Edit FRD  
 Low Priority:  
 e. Administrator Panel  
 f. Email notifications

4. Starting with the most important modules, write their specific requirements and prepare the SRS.

5. At the same time, prepare the architecture of the system and overall design and write the SDS. Also, prepare the designs for the important modules to be handed over to the programmers using the Requirements made by the Requirements Engineers.

6. When the requirements and design of the important modules are over, start the development of the most important modules.

7. Begin the development of the next modules by the time the testing of the already developed modules is over.

8. The steps above are repeated until development of all modules is complete and the system is integrated.

9. The system as a whole is verified and validated and then deployed.

10. Updates and Maintenance to the system is periodically done.

## 5.3. Algorithms

Some of the logics implemented in our Change Request Management Web Application are explained in the form of algorithmic pseudo-codes.

**Algorithm 1: Login**  
Begin  
if (User enters email && password && clicks login)  
then send login request to server  
if (email, password correct from database)   
then log user in and initiate user session for a time of 20 minutes  
else output “wrong email/password”  
End

**Algorithm 2: New Request**  
Begin  
if user chooses (Create New Request && fills form && clicks submit)  
then   
{ send FRD to the manager for approval, which appears in “Managed Requests” in the manager’s account  
Manager Received a notification informing him of a New Request pending approval/rejection  
save FRD in the Pending Requests section of the FRD Owner  
if (Manager clicks “Approve” New Request)  
then  
{ Status of the FRD changes to “Active”  
send FRD to list of employees in Panel – 7 for V.1 Approval/Rejection, which appears in their  
 Received Requests in a user’s account  
Move the Request to the Active Requests section of the FRD Owner }  
else if (Manager Rejects the New Request)  
then {  
Manager will be asked to provide a Rejection Note explaining the reasons of rejection  
Status of the FRD will be changed to “Closed”  
FRD Owner will receive a notification informing him of the Manager’s Rejection  
FRD will be moved to the closed section of both the owner and the manager } }  
else if user chooses (Create New Request && partially fills form && clicks save draft)  
then save New Request entered information to be completed later  
else do nothing  
End

**Algorithm 3: View Closed Request**  
Begin  
if (user chooses to View all closed requests)  
then { retrieve all closed requests from database  
View all closed requests as a clickable list on the page, sorted by the latest closed requests }  
if (user clicks on a request)   
then view that request’s information  
End

**Algorithm 4: Edit/View Active Requests (Only for the FRD Owner)**  
Begin  
if (FRD Owner chooses to Edit/View all Active requests)  
then { retrieve all active requests owned by the user from database  
View all current active requests as a clickable list on the page sorted by the latest updated version }  
if (user clicks on an active request)  
then view that request’s information  
if (user edits information and does not click update Request)  
do nothing  
if (user edits information and clicks update FRD)  
then { create a new version of the FRD, update contents of FRD in database  
notify Panel – 7 list of employees with a new version to be accepted }  
End

**Algorithm 5: View/Update Received Requests (For Panel – 7 FRD Users)**  
Begin  
if (user logged in && user receives a new request as part of Panel – 7)  
then { display notification of new available request  
display a new received request in Received Requests interface }  
if (user clicks on Request in Received Requests)  
then view that request (Or view another Version of the request)  
if (User approves the latest version) //User may only approve the latest version of the FRD  
then { mark that request as approved by the user  
notify the FRD Owner of that user’s approval }  
else if (User declines the latest version)  
then { mark that request as declined by the user  
notify the FRD Owner of that user’s rejection }  
else if ((User submits a new comment OR new file) AND Clicks Submit Update)  
/\*The user may submit a new version of the FRD even without approving or rejecting the previous version \*/  
then {  
create a new version of the FRD, update it in the database  
notify the FRD owner and all the users of Panel – 7 about the new version }  
else  
do nothing  
End

## 5.4. Standards

Some of the Standards implemented for our Change Request Management Web Application Project are as follows:

Project Proposal Form: TUBITAK

Software Requirements Specification Document: IEEE 830-1998

Software Design Specification Document: IEEE 1016-1998

Security: SHA256 Hashing with Salt

Implementation: Standard ECMA-262 8th Edition - June 201723

## 5.5. Detailed description of the implementation

As the design and requirements step for each module is complete, the implementation step for that module begins. In this section, we will consider the implementation of the Frontend, the Backend, the Web API and the Database. The application backend was developed mainly using the C# Programming Language for both the server side and the Application API. The application frontend was developed mainly using HTML, CSS, JavaScript and various frontend frameworks and libraries such as React, JQuery, Bootstrap. There are more than 20 design pages (.cshtml pages using the razer engine) in the implementation of the code on Visual Studio. There is only a single layout file (layout.cshtml) that acts as the basis of all design pages (including the sidebar, header, footer).

**Frontend Implementation**

For the Frontend of our Web Application, we used the Bootstrap Studio tool to generate some User Interface (UI) code and assist us with the frontend process. We also implemented the User Interface using Bootstrap CSS and JavaScript libraries for the aesthetics and styling of the HTML markup. The SaSS CSS library was also used to further style the Web App and to make it look modern and user friendly. We used a bundle called the Webpack which allows us to import and export JavaScript libraries and files. This is used specifically for the implementation of the JavaScript React framework. The react framework was used in the development of several components within the web application. React is mainly used for the responsive rendering of HTML Components using JavaScript and JSX (which is a markup written in JavaScript files but gets executed or rendered as if it was HTML Code). Additionally, a compiler called Babel was used in the project in order to facilitate the compilation of JavaScript ES6 syntax to ES5 syntax since most browsers nowadays do not support the new JavaScript ES6 syntax yet. This react component was mainly used in Panel 4 (Add new SMS, Import SMS). In addition to React, we also used the JQuery JavaScript library in various parts of our frontend. The React files responsible for the SMS implementation in Panel 4 are shown in figure 25.

For the implementation of exception handling, we implemented various HTTP Errors exception handling where error responses from the server such as error 404 or error 503 are handled with meaningful error messages presented to the user. When an error occurs due to several different reasons such as requesting a non-existing page in the server, an error with code 404 is shown. Also, if an internal error occurs, such as a bug in the code or in the database, an error message with code 503 with a relevant meaningful message explaining the error is displayed. This is shown in figure 26.

For the implementation of the filtering and sorting functionality, we used the List.JS library which is a frontend open source JavaScript library to assist us in this process. Using this library, we were able to sort all the Active, Closed and Received FRD Lists by their FRD ID, Name, Date Created, Date Updated, Date Closed. We will not use the List.JS library for the universal search feature of our Web Application.

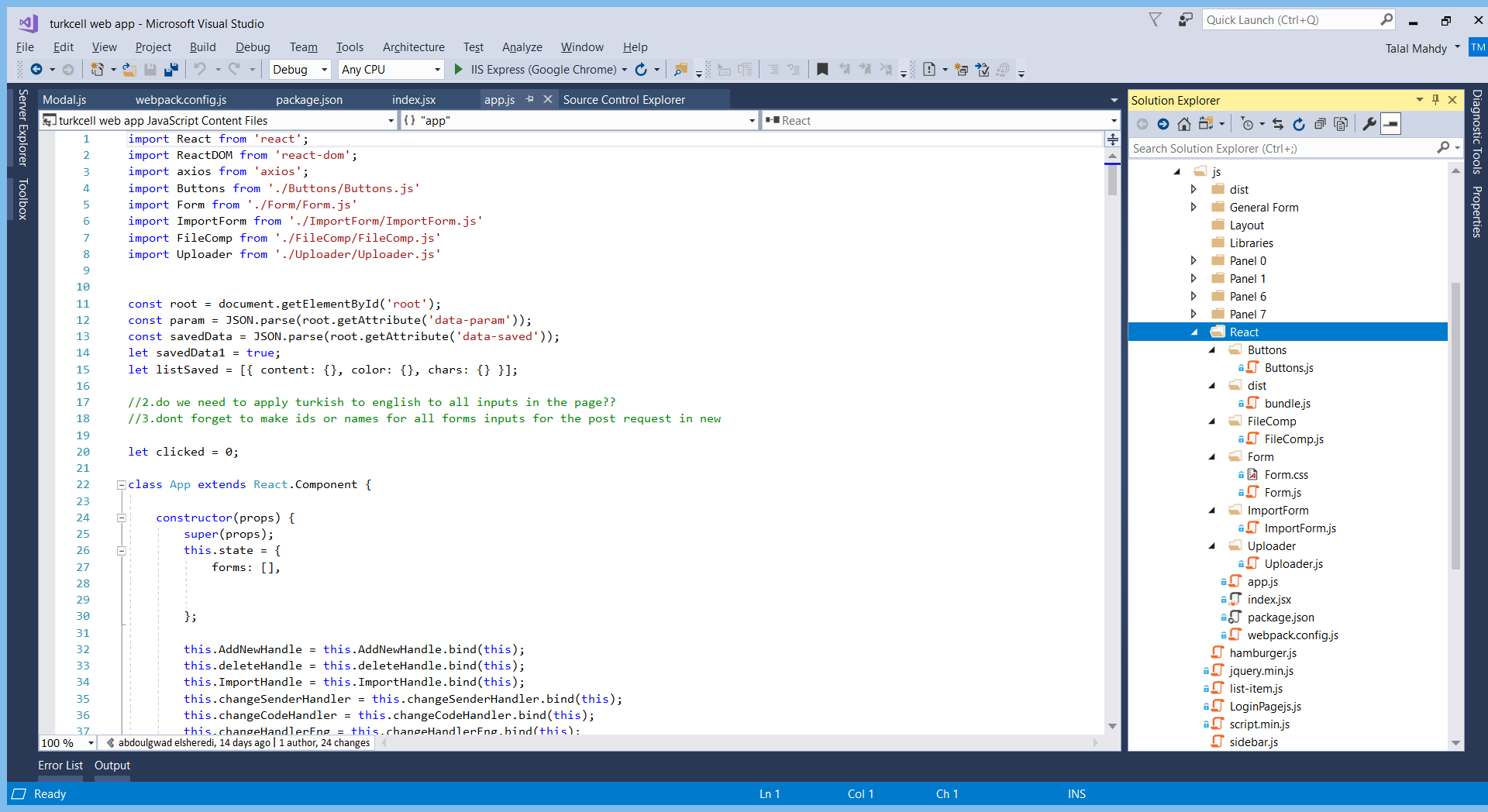


Fig. 25 React Files frontend Implementation

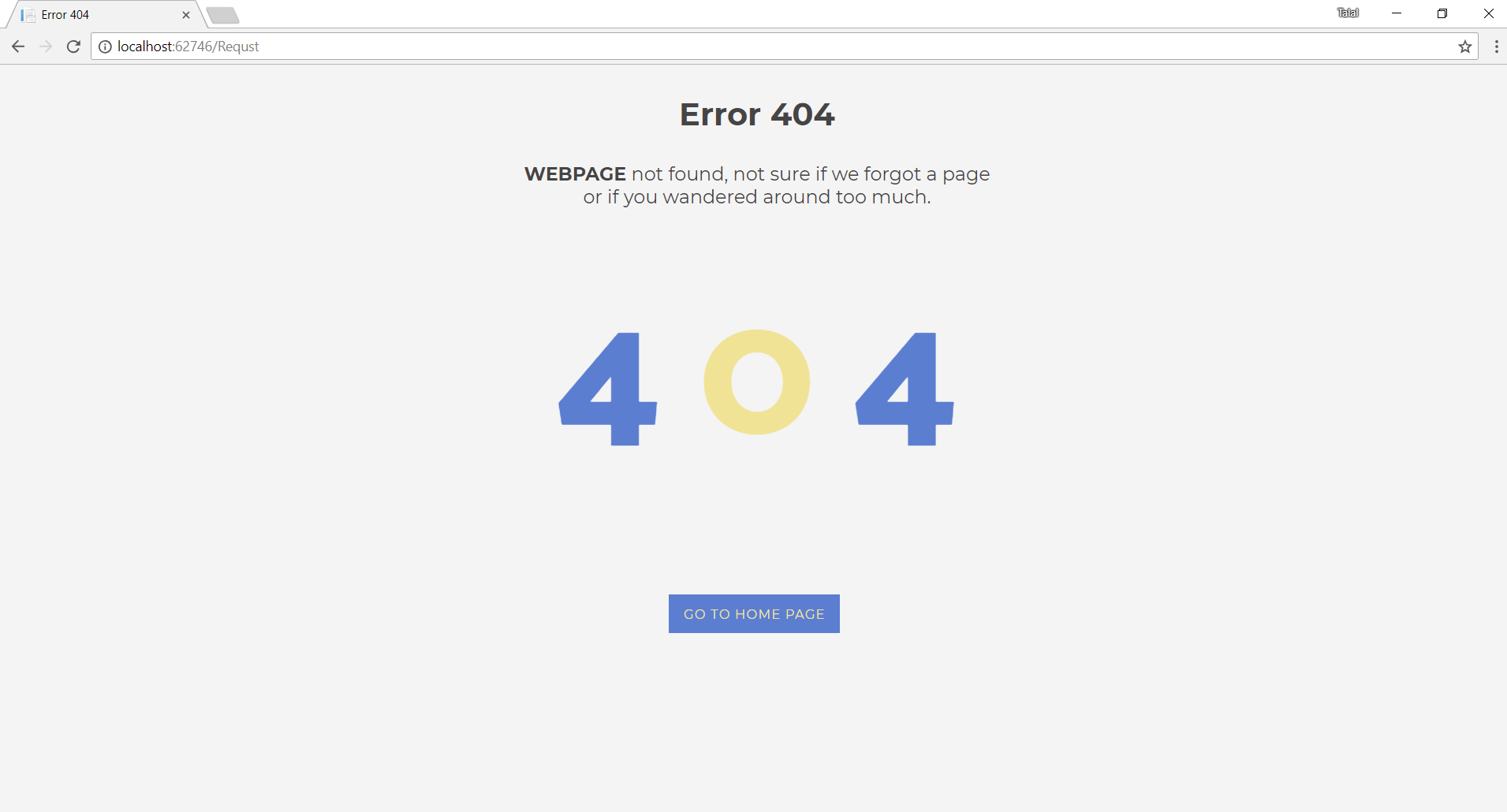


Fig. 26 Error 404 Implementation

**Web API Implementation:**

With regards to the API, ASP.NET RESTful API was used in several parts of the application in order to facilitate AJAX requests between the database and some components in the application such as Panel 4’s SMS codes and SMS contents retrieval in case the user decides to import previously saved SMS data from the database. In addition, the API is used in Panel 0 of the new request page in order to check if the inserted FRD Number is already used or not in real time.

**Backend Implementation:**

The backend section of our Web Application consists of both the Model and the Controller Sections of the MVC Architecture.

In the Model section of the MVC architecture in the web application, classes are divided into several sub-categories. The model classes of our application are shown in figure 27.

1. Business logic models: Represents the business rules derived from the Requirements Specification Document. It is specifically designed with the principles of loosely coupled development (Trying to separate different layers of the application) in order to allow expandability and reusability of the application. Example: Employee.cs (Reviser.cs and Manager.cs inherits from Employee.cs), SMS.cs, Department.cs, Notification.cs, Email.cs, Comment.cs, etc. The important Employee.cs business logic model class is shown in figure 28.

2. Presentation logic view model: Represents classes of data that are displayed on specific views or pages. Example: Frd.cs, NewFrd.cs, Credentials.cs, ReceivedFRDs.cs, NotificationPage.cs, etc. The important NewFrd.cs presentation logic view model is shown in figure 29.

3. Database implementation functions: These models are used for the processing of the database functions throughout the project. These processes include connecting, querying (implemented in data sets), and updating the database using Oracle commands and adapters. The best example of this type of class is the DB\_Functions.cs class. This class contains over 73 functions which are responsible for all the connections and insertions of data to the database. This class is shown in figure 30. Another important database implementation class is the G\_Functions.cs class.

The controller section of our MVC Web Application acts a medium facilitator between the View, which is related to the frontend of the application, and the Model which is related to the backend of the application. The important implemented controller classes in our application are RequestController.cs, NotificationsController.cs, HomeController.cs, FileController.cs, etc. The important RequestController.cs controller of our application is shown in figure 31.

For the universal search feature implemented at the top of the pages of the Web App, we should write backend C# Code as we are going to retrieve the requests from the database using SQL. We will be able to search for all the Requests that the user can access using the FRD ID and the FRD Name.

For the security of our Web Application, we implemented both Server side and Client side validation for all the forms in our application. We implemented Server Side Validation to protect against malicious users who can potentially submit dangerous inputs to the server. We implemented Client Side Validation in case a user submits a wrong input by mistake such as submitting a wrong Email or a wrong FRD ID. We also implemented SHA-256 Hashing with Salt for encrypting the login password in the database.

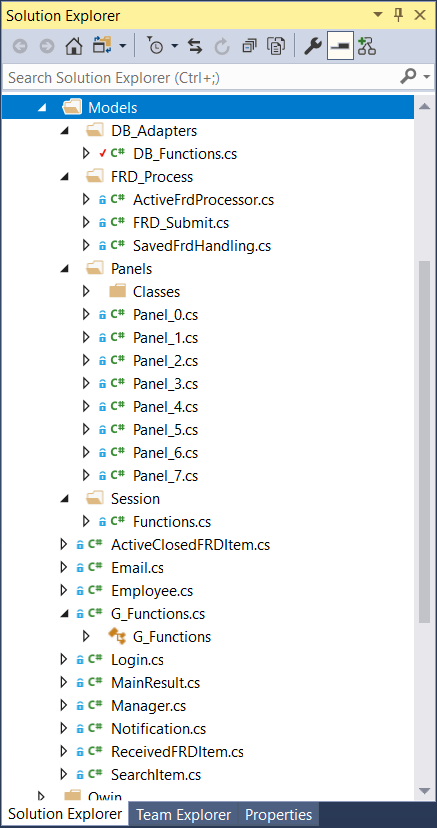


Fig. 27 Model Classes

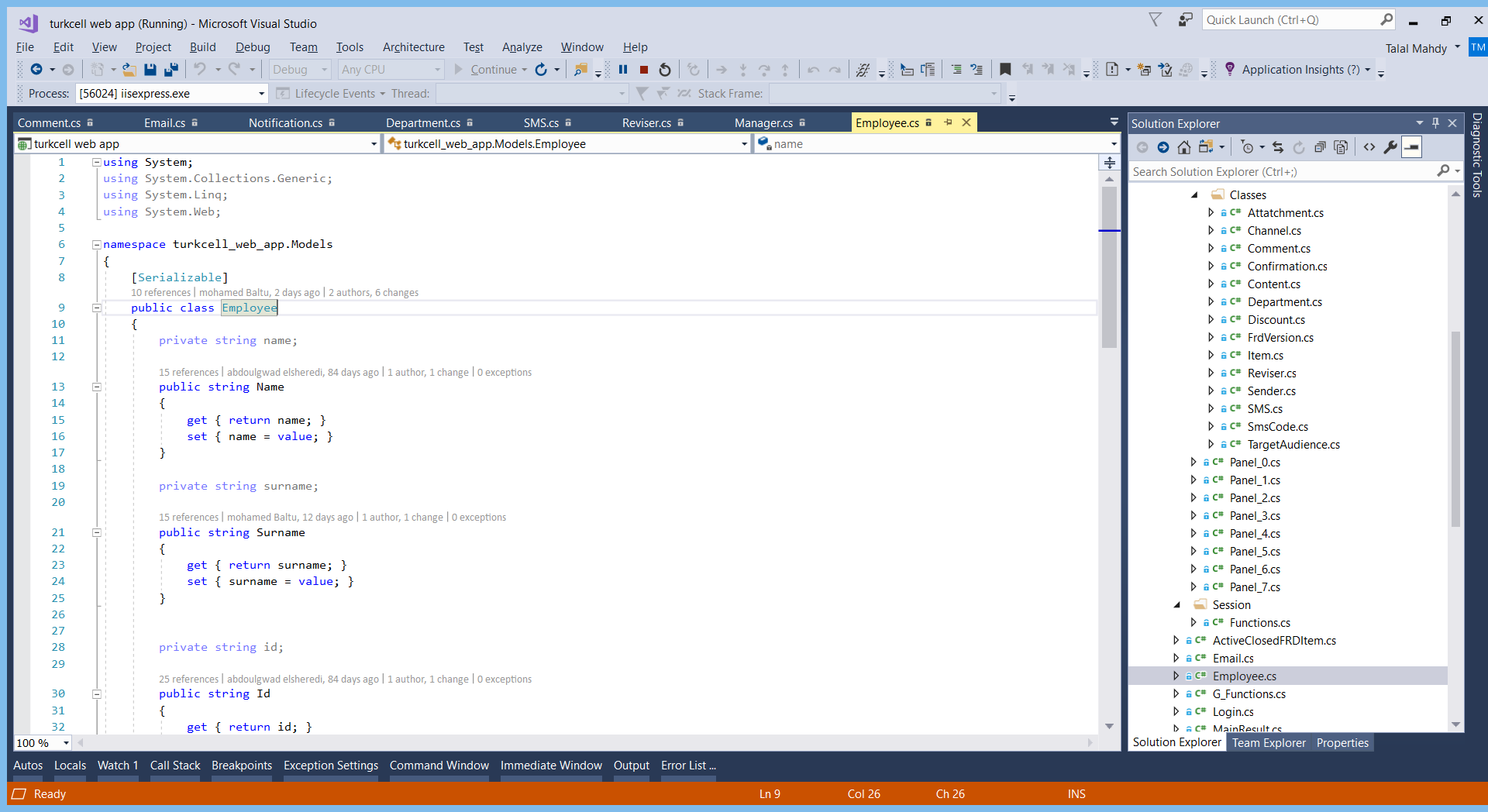


Fig. 28 Employee.cs business logic Class

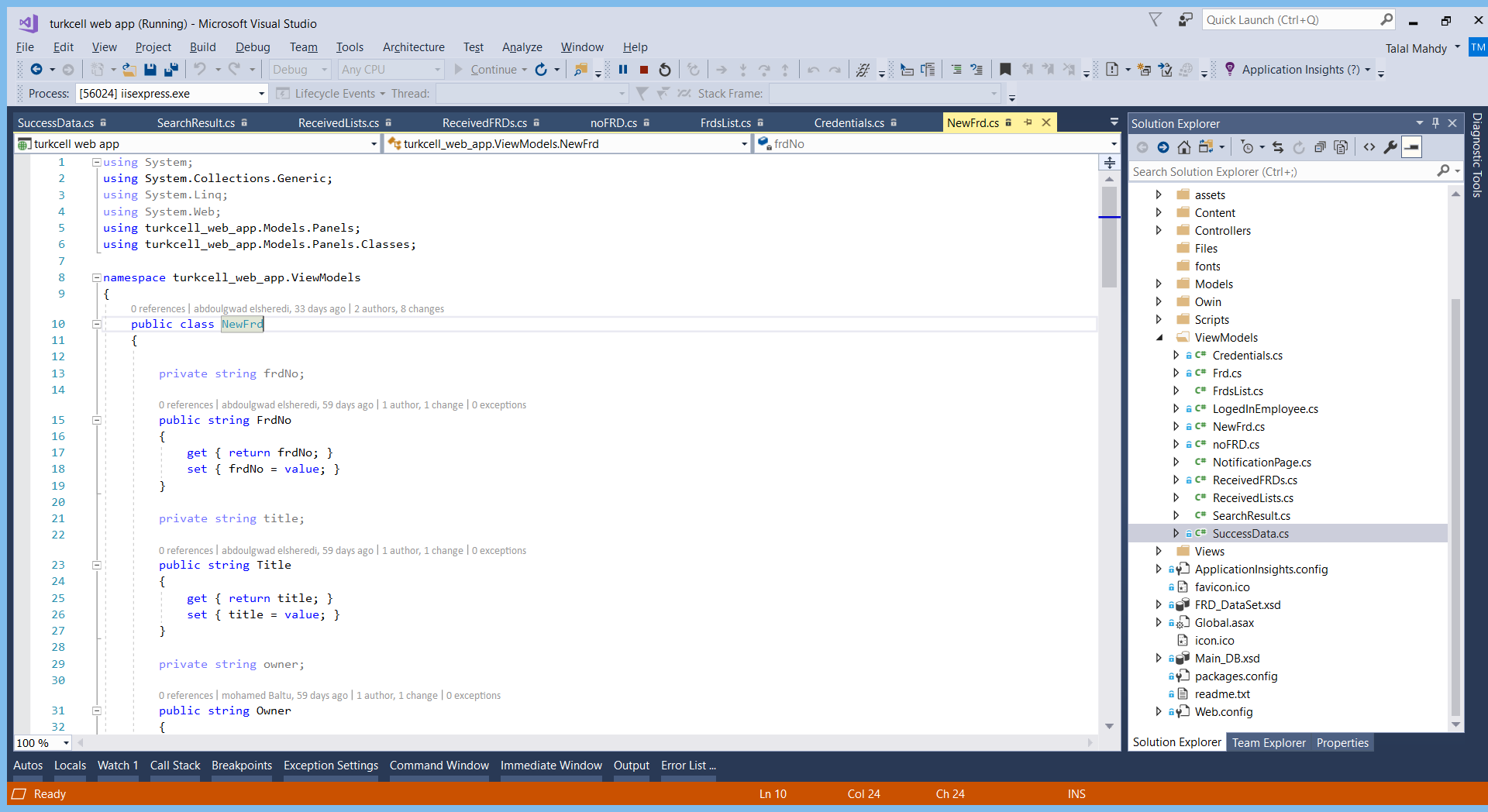


Fig. 29 NewFrd.cs Presentation logic ViewModel Class

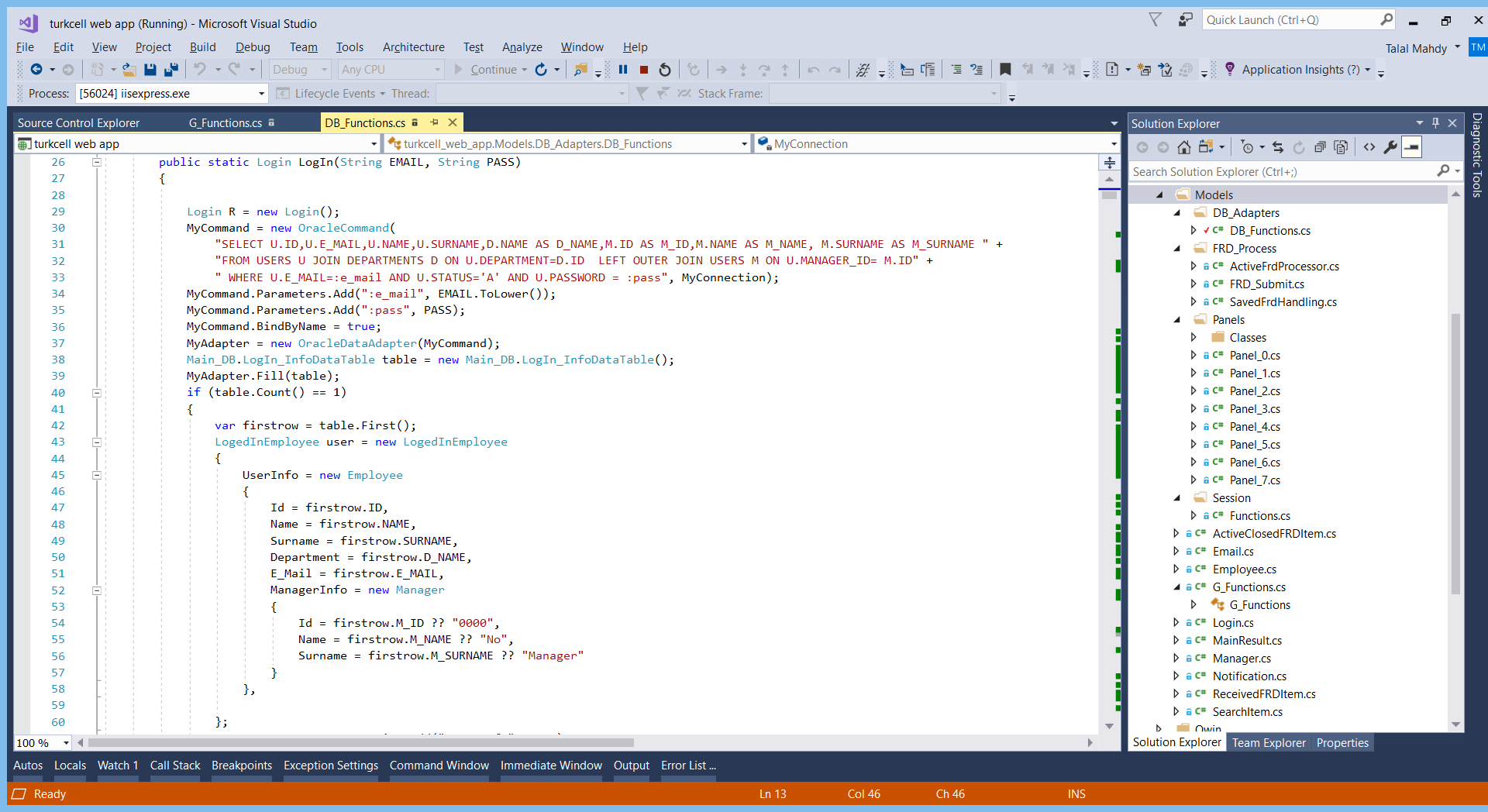


Fig. 30 DB\_Functions.cs Database Implementation Class

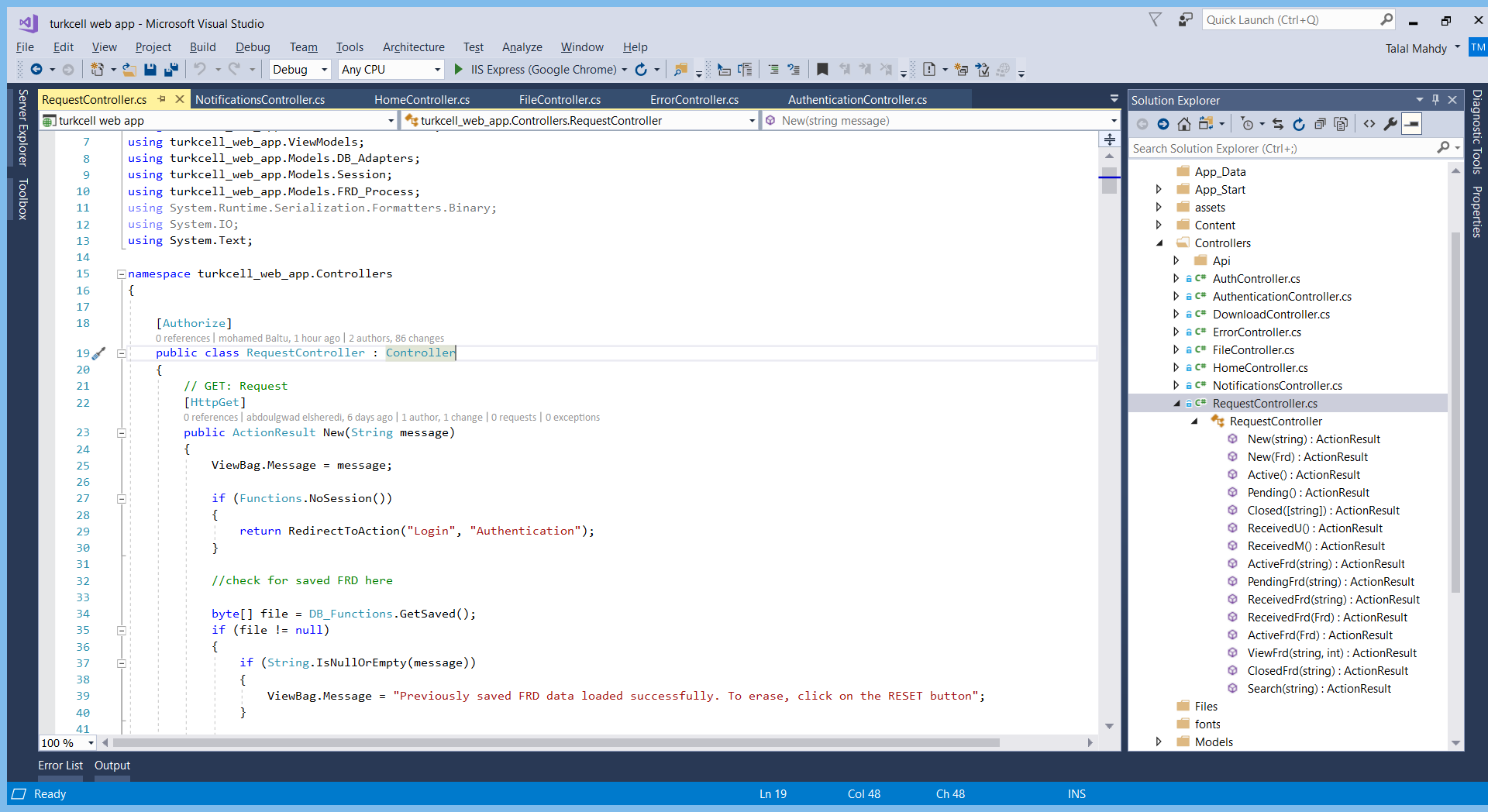


Fig. 31 RequestController.cs Controller Class

**Database Implementation:**

There is only one locally hosted database for the application which was implemented using Oracle. An Oracle database differs from Microsoft SQL Server in that some of the SQL syntaxes differ. Also, some of the main data types such as Boolean and bits do not exist in Oracle. There are 25 tables in the database implementation on Oracle which are shown in figure 33. A procedure is a function that does not require any output. We implemented 3 different procedures in the database which are New\_Confirmations, Notify\_Frd, Update\_Latest\_Version. To facilitate various functions of the web app and to easily update the tables required in the database, we implemented 3 various trigger functions in the database. A trigger function is activated whenever any change in the database happens. The Frd\_Activated trigger function allows the FRD to be shown to all the users of Panel – 7 after the manager accepts the FRD. The two other trigger functions implemented are Inserted\_Version and Frd\_Activated. These procedures and triggers are shown in figure 32. All the Database related information in included in a special class called the DB\_Functions.cs which is contained in our Visual Studio Web Application. The connection string implemented contains information such as the Database name, data source IP Address, username of the database, password of the database.

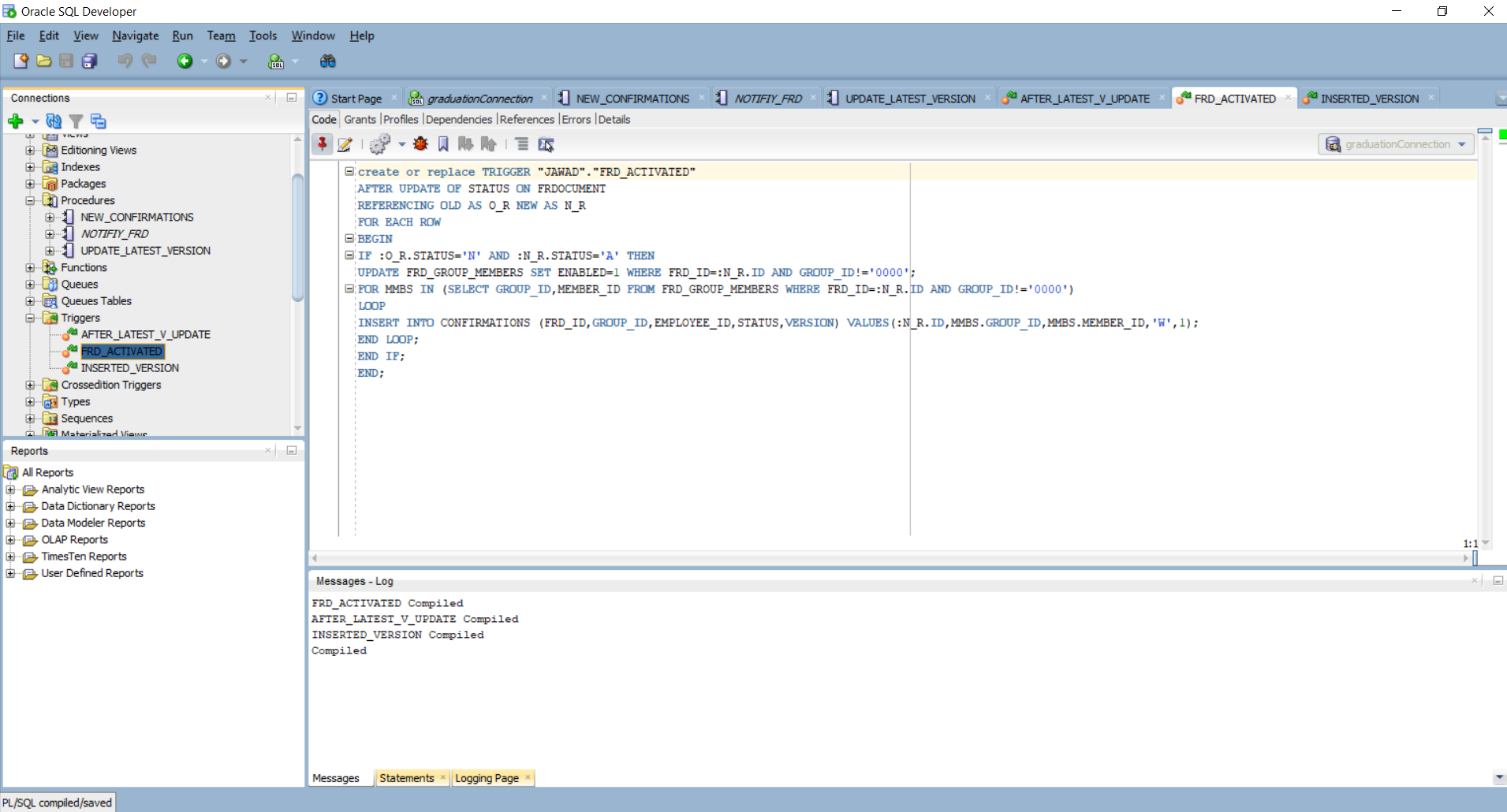


Fig. 32 FRD\_ACTIVATED Trigger function in Database

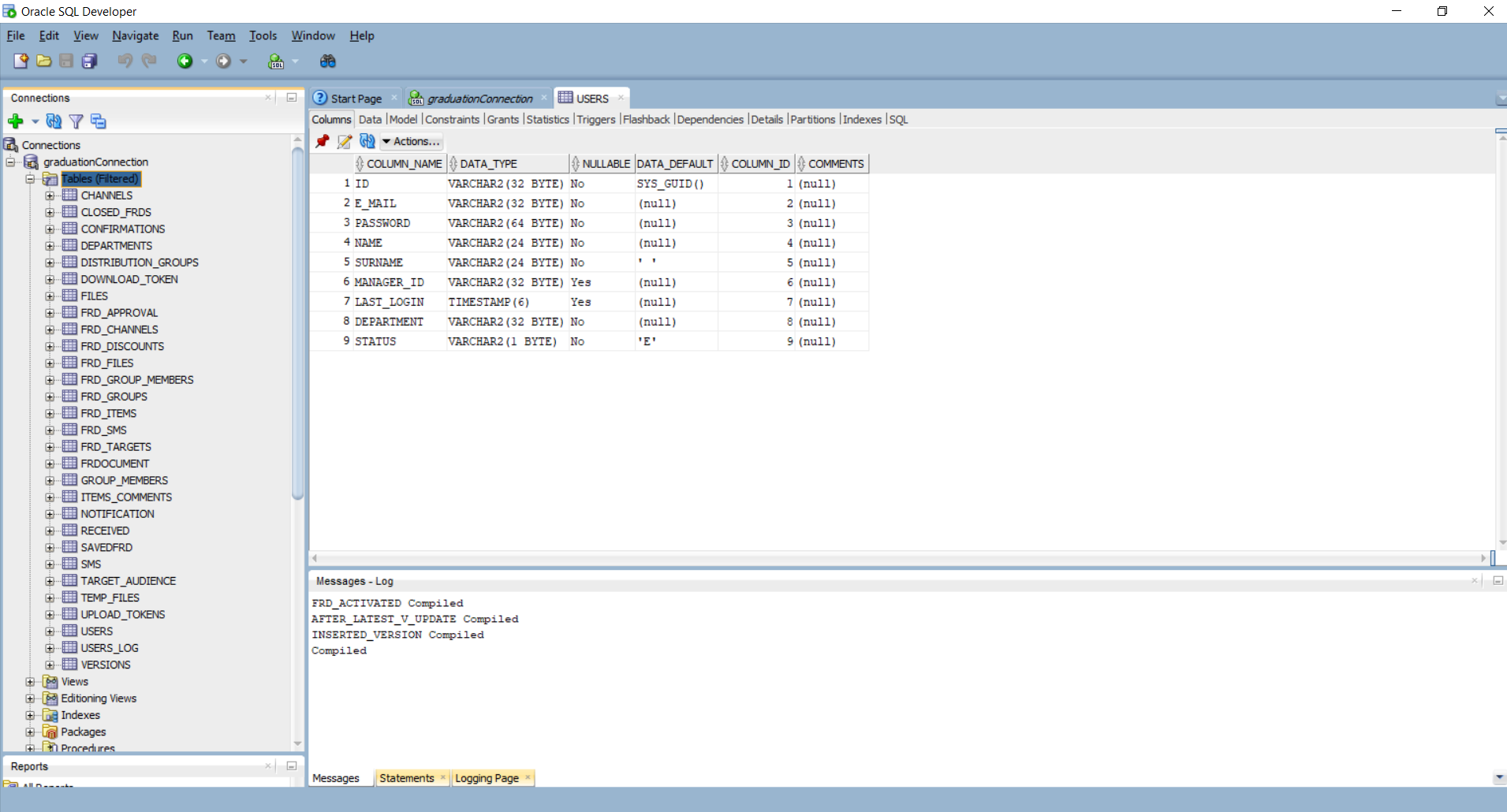


Fig. 33 Database Tables in Oracle

**Login Implementation:**

When the user runs the application, the first page shown is the login page and if the user logs in, the application makes sure that a connection between the login functions and the database is established. After a user clicks the login button, the browser of the user sends an HTTP post Request containing the login credentials to the server hosting the web application which redirects the user data to the database using the controller function which exists within the ASP.NET MVC Framework. Then the web application receives the information and forwards it to the Database to check if the database contains any user with the same email and password and the application performs validation and verification of the user data. Then the web application determines whether the user is able to log in based on the response of the database and stores the user ID on the session storage which will be used to process any future requests until the user chooses to sign out or the session key expires (after 15 minutes of inactiveness). An important login class used in our web application is the LogedInEmployee.cs class. This class is shown in figure 34.

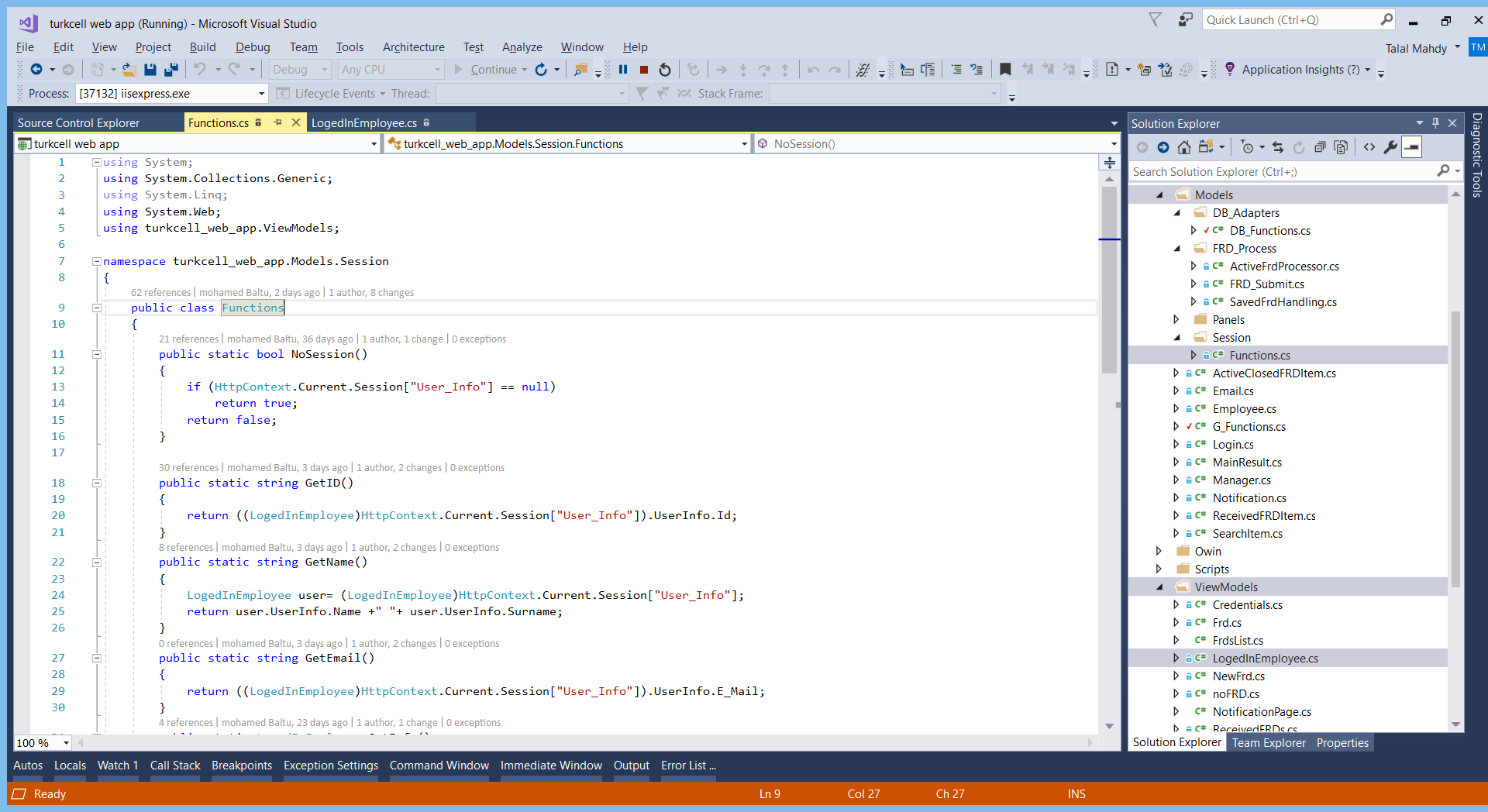


Fig. 34 Login functions Functions.cs and LogedInEmployee.cs

# 6. TESTING

Modules of the application were tested manually for any bugs, crashes as well as any unexpected output results. Expected outputs were compared with the outputs perceived during the testing and subsequent changes and modification to the code were made in case of mismatch. The validity of the end product has been verified by trying the web application on several browsers such as Google Chrome and Mozilla Firefox with different specifications in order to assess visuals, resolution, colors and compatibility of the web application. This method also allowed the use of these browsers to test the app during its development, not only to validate the end product. Testing results showed an overall good standing of the end product with major features of the application working smoothly without crashes or unexpected outputs. In addition, some older browser versions showed no compatibility with some ES5 JavaScript features of the application. However, some minor bugs might be present that require further testing after the end product release due to the large number of variables that might affect the application’s various features and functions.

Table 14. Test Case 1

|  |  |
| --- | --- |
| Test Case ID | TC-01 |
| Test Case Name | Login test |
| Pass/Fail Criteria | Pass: user enters correct email, password -> login  user enters wrong credentials -> “wrong email/password” |
| Input Data | Email address with a proper format  Numeric/Alphabetic/Symbolic Password |
| Test Procedure | Expected Output: |
| Step 1: Enter correct email, password | Logs in to system |
| Step 2: Enter wrong email/password | Displays a message “wrong email/password” |
| Comments | Test passes successfully |

Table 15. Test Case 2

|  |  |
| --- | --- |
| Test Case ID | TC-02 |
| Test Case Name | Create New FRD Test |
| Pass/Fail Criteria | Pass: a new FRD report with version 1 successfully created, sent to manager for approval.  Fail: new FRD fails to be created or fails to be sent to manager. |
| Input Data | FRD Report inputs (Panel 0 + Panel 1 + ... + Panel 7) |
| Test Procedure | Expected Output: |
| Step 1: Login to user account. Create New FRD. Write FRD required details (Request Number, Request Name, minimum 1 description, minimum 1 confirmation) and submit FRD. | FRD successfully submitted and saved in database. FRD Status is set as pending which means that the FRD is pending the manager’s approval to proceed with working on it. |
| Step 2: Login to the manager’s account. | Manager gets a notification notifying of a new pending request awaiting approval. |
| Step 3: Click on “Received as Manager” button | Displays all received FRD’s awaiting acceptance from the manager. |
| Comments | Test passes successfully |

Table 16. Test Case 3

|  |  |
| --- | --- |
| Test Case ID | TC-03 |
| Test Case Name | Approve/Rejecting FRD as a User or as a Manager |
| Pass/Fail Criteria | Pass: After clicking on “Received Requests as User” or “Received Requests as a Manager”, we shall be able to open any FRD version and approve/reject it, after which the request owner shall be notified of the approving/rejection decision.  Fail: Web Application crashes when approving/rejecting or the owner does not get notified. |
| Input Data | Opening Received FRD’s, pressing approve/reject buttons |
| Test Procedure | Expected Output: |
| Step 1: Login to user account. Create and submit an FRD. Login to manager account. | FRD successfully submitted and saved in database. Manager gets a notification about a pending decision. |
| Step 2: Open the “Received Requests” sidebar dropdown link, click on “Received as Manager”. Open any received FRD from the list. | Manager is able to open any FRD from the list and will see an approval/rejection to be made by him. |
| Step 3: Try accepting an FRD from the Received as Manager list. Login to the user’s account again. | The user will get notified about the manager’s approval and may begin working on the FRD with the list of Panel – 7 employees. |
| Step 4: Try declining an FRD from the Received as Manager list. Login to the user’s account again. | The user will get notified about the manager’s rejection, along with a possible rejection message explaining the reasons of the rejection. The user may not start working on the FRD. The FRD status changes to closed and goes to the list of “Closed Requests”. |
| Step 5: Login to a Manager’s account. Create an FRD, add some people from Panel – 7’s list. | The FRD’s status is automatically changed to active and work on the FRD shall start immediately since the manager is the one who is creating the FRD this time and there is no one appointed above him. |
| Comments | Test passes successfully |

Table 17. Test Case 4

|  |  |
| --- | --- |
| Test Case ID | TC-04 |
| Test Case Name | Collaborate on an FRD after the manager’s approval |
| Pass/Fail Criteria | Pass: Two or more team members from Panel – 7’s list are able to properly comment and collaborate on an approved FRD.  Fail: Two or more team members from Panel – 7’s list are not able to properly collaborate on an approved FRD. |
| Input Data | Newly written comments/uploaded files by other team members from Panel – 7 |
| Test Procedure | Expected Output: |
| Step 1: Create a new FRD as a user. Choose two additional teammates in addition to the manager from Panel – 7’s list. Get it approved by the manager. | FRD successfully submitted and saved in database. Manager approves the request. Everyone gets notified. Work starts on FRD. //Note that the manager is also an employee in this FRD request //since we chose him from Panel – 7’s list. Had we not chose him, //the manager’s job would have been to only approve/reject the //FRD. |
| Step 2: Login to the first teammate account. Click on “Received Requests”, “Received as User”. Choose the FRD we created. Open panel 1, write a new comment. Open panel 5, upload a new file. Click Update. | A new FRD version is created. The manager and panel – 7 employees receive a notification regarding a new version of the FRD to be approved/rejected. The old and new FRD versions appears in the Received as User section in the both the teammates accounts and the manager’s account. For the request owner, all versions appear in the Active Requests section. |
| Step 3: Accept the FRD from the owner, second teammate and manager accounts. | The FRD version status does not get changed to “Closed” since only the request owner may close the request, even if all his team members accepted FRD Version. |
| Step 4: Go to the FRD owner’s account, “Active Requests”, click on the FRD and close it. | The FRD version status is changed to “Closed”. It gets moved to “Closed Requests”, “My FRD’s” section for the request owner. For all other team members and the manager, it gets to moved to the “Closed Requests”, “Other’s FRD’s” section. |
| Comments | Test passes successfully |

Table 18. Test Case 5

|  |  |
| --- | --- |
| Test Case ID | TC-05 |
| Test Case Name | SHA-256 Hashing with Salt Security Test |
| Pass/Fail Criteria | Pass: User passwords are saved in a hashed form in the Oracle Database  Fail: User passwords are plainly saved in the Database |
| Input Data | Database connection, Users table inspection |
| Test Procedure | Expected Output: |
| Step 1: Open the Oracle Database and connect to the database. | Successfully connects to the database server. |
| Step 2: Visit the Users table, check if the passwords are hashed. | Passwords appear to be hashed and secure in the database. //Note that Man-in-the-middle attacks may still be possible since //the password is hashed only after reaching the database. To //implement HTTPS, an SSL Certificate is required from a //Certification Authority (CA). |
| Comments | Test passes successfully |

Table 18. Test Case 6

|  |  |
| --- | --- |
| Test Case ID | TC-06 |
| Test Case Name | Responsiveness Test |
| Pass/Fail Criteria | Pass: Website is dynamically responsive to any screen size or resolution.  Fail: Website remains static and does not react to any reduction in the browser window size or resolution. |
| Input Data | Increase/Decrease browser window size |
| Test Procedure | Expected Output: |
| Step 1: Run the Web Application in Google Chrome, FireFox, IE, Microsoft Edge Browsers. Maximize the window size. Gradually decrease the window size and watch if the components react to the change in size. | The Web Application successfully runs on all major web browser. The web application is responsive and it reacts positively to any change in the browser window size due to the use of the Bootstrap CSS Library. |
| Comments | Test passes successfully |

7. USER GUIDE OF THE SYSTEM

**Basic flow of the system:**

When the user launches the Web Application, the login page shown in figure 35 will appear providing input fields for Email and Password, as well as a login button. The user/admin registration feature is not available since it was not part of the requirements. Registration may only be done directly by the system administrator through the application’s database.

If a user or a manager logs in to the system, they will be presented with the following main webpage as shown in figure 36. Note that a user account will not be having the Manager Settings option in a future update to the system.

On the top of the page, we have the header of the page which contains the logo of KKTCell, a universal search bar (search by FRD ID or by FRD Name), a notification drop-down list button, and a user account button (Shows account details and a logout button). On the left side of the main page, we will find the sidebar which contains a link to the Home Page, all the main functions of the web application and a logout button. On the right main side of the page, we will find the Dashboard for the logged in user. The dashboard shows all the details about the user as well as insightful statistics on the user’s history with the application, such as the number of Active FRD’s he currently has and the number of FRD’s he has ever received. The dashboard will also show whether the logged in user is an administrator or not.

**Active FRD’s Guide:**

Only the Request Owner’s initiated requests will be places into this FRD List.

User: If a user creates a new request, the manager would have to approve it first. Therefore, the status of the newly created request is “Pending”.

Manager: If a manager creates a new request, the newly created request’s status is automatically set as “Active”.

**Pending Requests Guide:**

This list is needed for all the user request owners who are waiting for their manager’s approval to start working on a request.

**Closed Requests Guide:**

Only the Request Owner can close a request, regardless of whether the list of panel 7 employees accepted or rejected the request. In the closed requests sections, we notice that there is a section called “My FRD’s” and another section called “Other’s FRD’s”. Since only the request owner may close a request, we added another section for the requests closed by another request owner while we were a panel 7 team member of the same request.

**Received Requests Guide:**

User: If employee 1 creates a new request (Status must be active, or just changed from pending to active) and adds employee 2 to the panel 7 list, then employee 2 will receive a new request with version 1 ready to be approved/rejected. Any subsequent version updates to the FRD will be received here. The “Received as Manager” section is where the managers can approve/reject a newly created FRD.

**Panel Guide:**

Panel 0: Request Number: Required and must be unique  
 Request Name: Required

Panel 1: Request Description: At least 1 description comment is required. This is a collaborative  
 feature where other users may add extra comments to the FRD and create a new version.

Panel 2: You may choose a target audience (Optional).

Panel 3: You may choose various channels affected by the demand (Optional).

Panel 4: (Optional)   
 Add New: Request to add a new SMS row to the SMS table (Won’t actually add unless   
 it is manually done from the database)  
 Import: Request to change/delete an already existing SMS column.

Panel 5: You may upload any type of files by dragging and dropping them to the box or by   
 clicking on the box and choosing the files individually. This is a collaborative feature.

Panel 6: You may request to add new discount prioritizations to the table (Won’t actually add   
 unless it is manually done from the database) (Optional)

Panel 7: This is an important panel where you must pick at least 1 collaborative team member   
 who will approve/reject and work with you on the FRD.

Save to Draft: Saves the information written in textboxes and checkboxes for the user to complete them at a later stage.

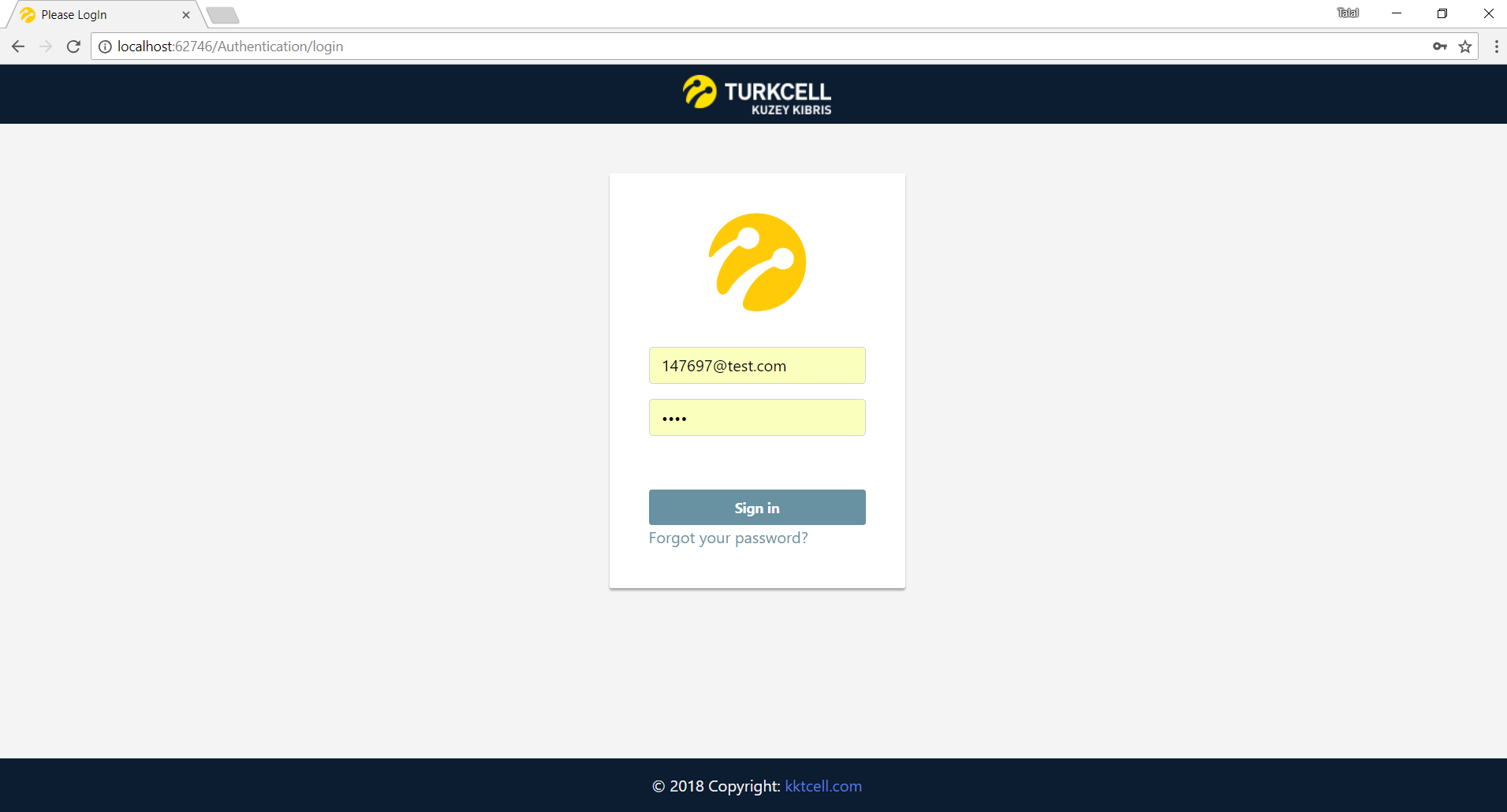


Fig. 35 Login Page

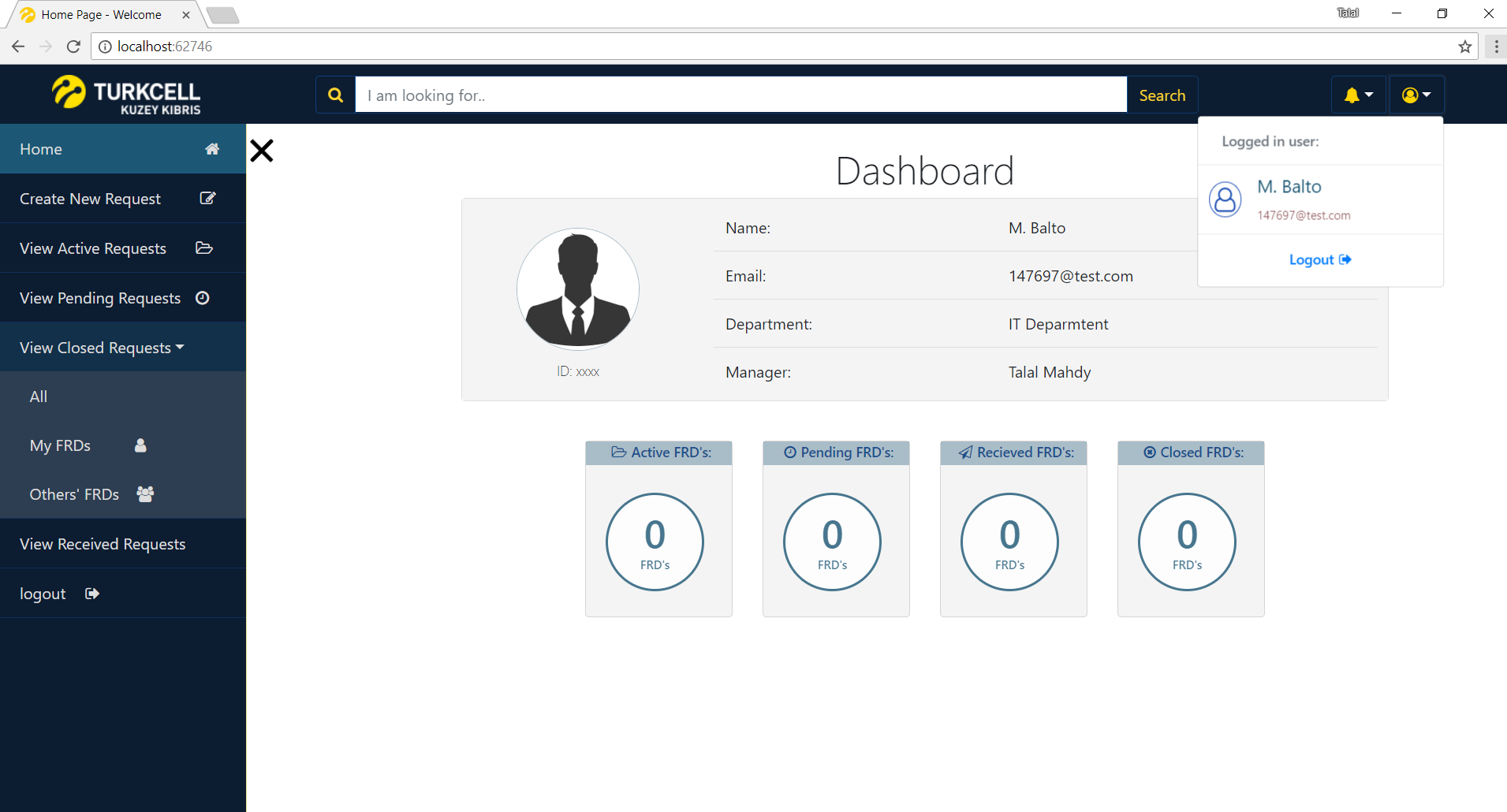


Fig. 36 Main Landing Page

# 

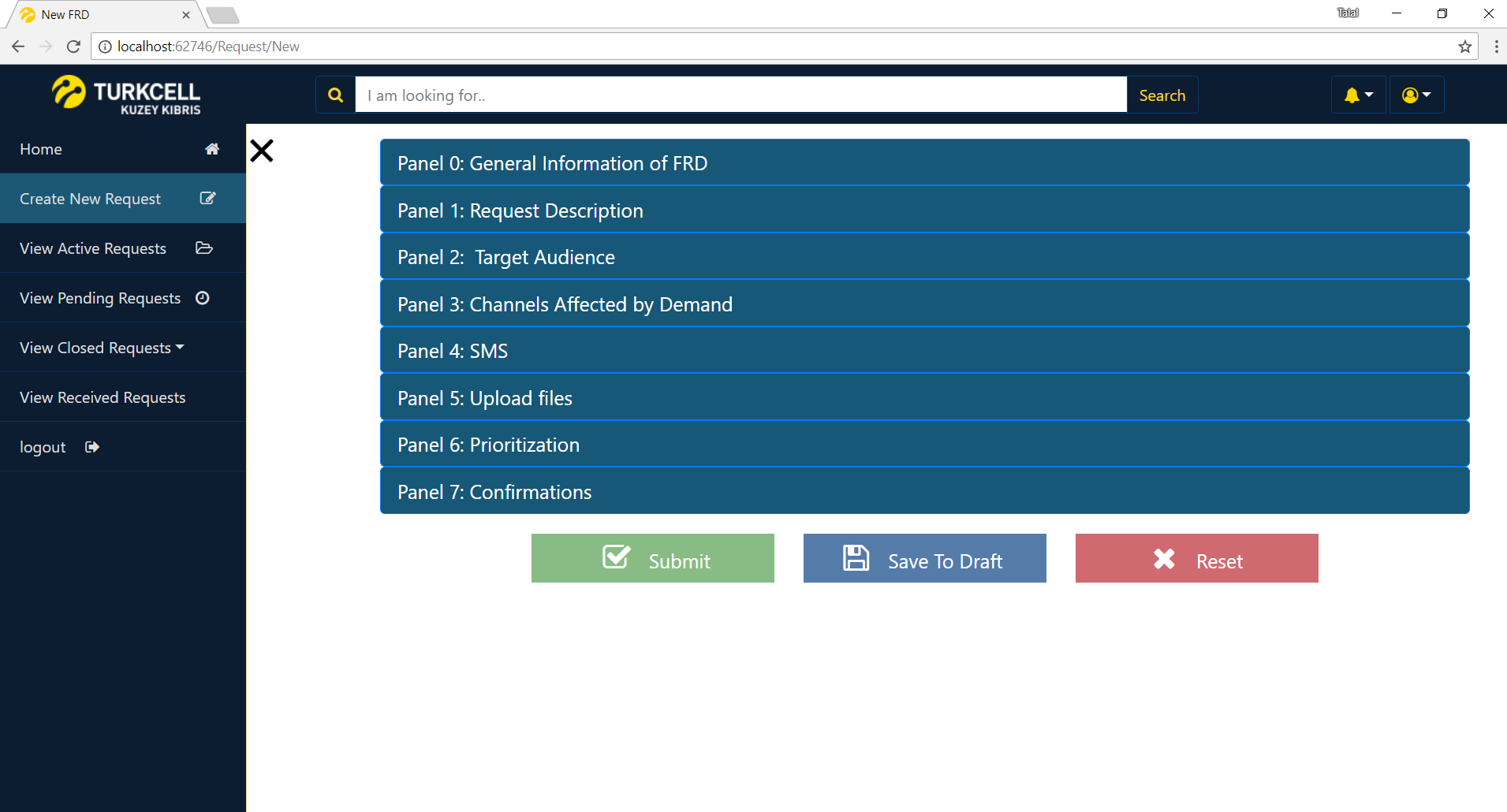


Fig. 37 Create New Request – Panel Page

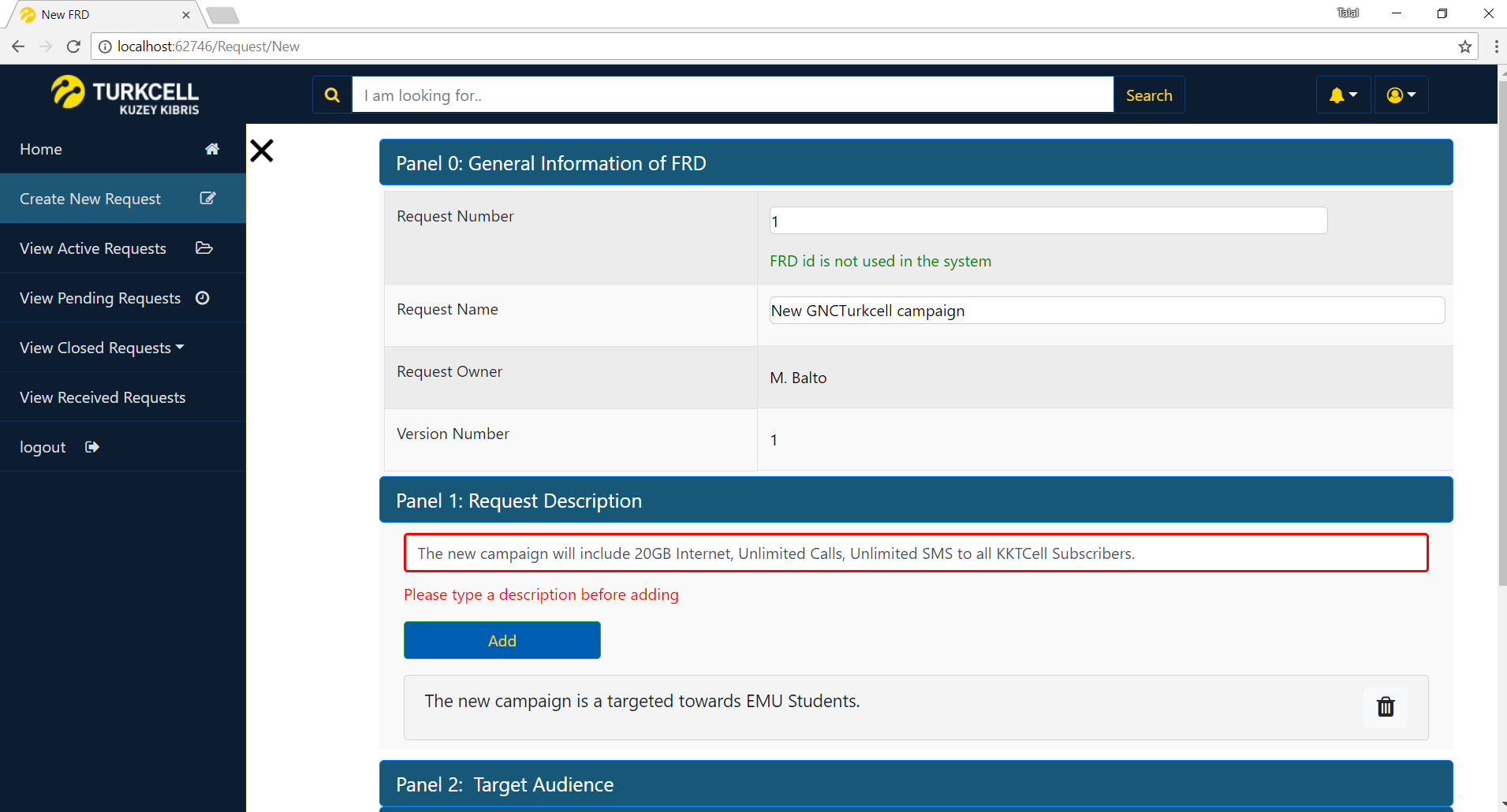


Fig. 38 Panel 0 and Panel 1

# 

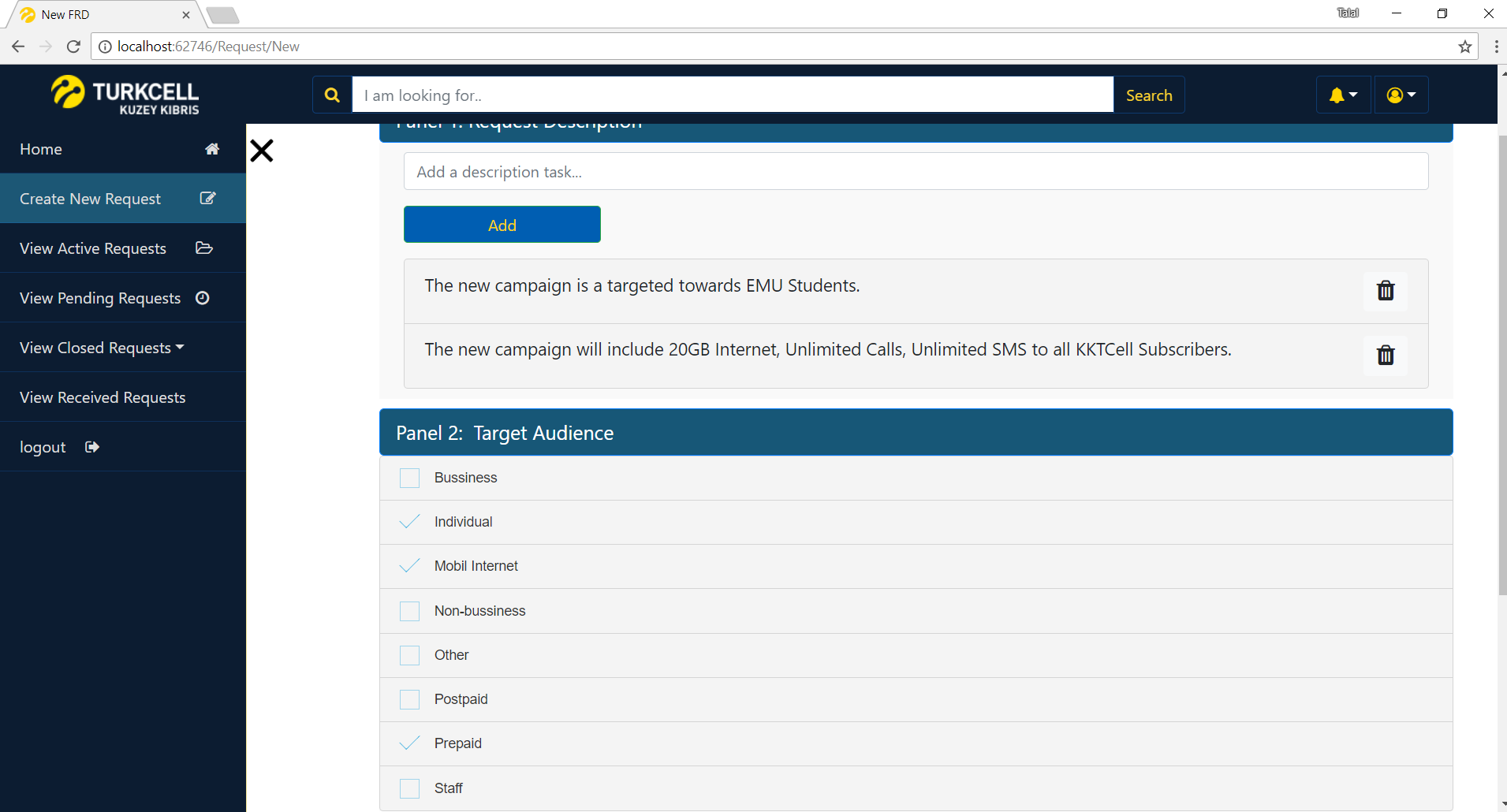


Fig. 39 Panel 2

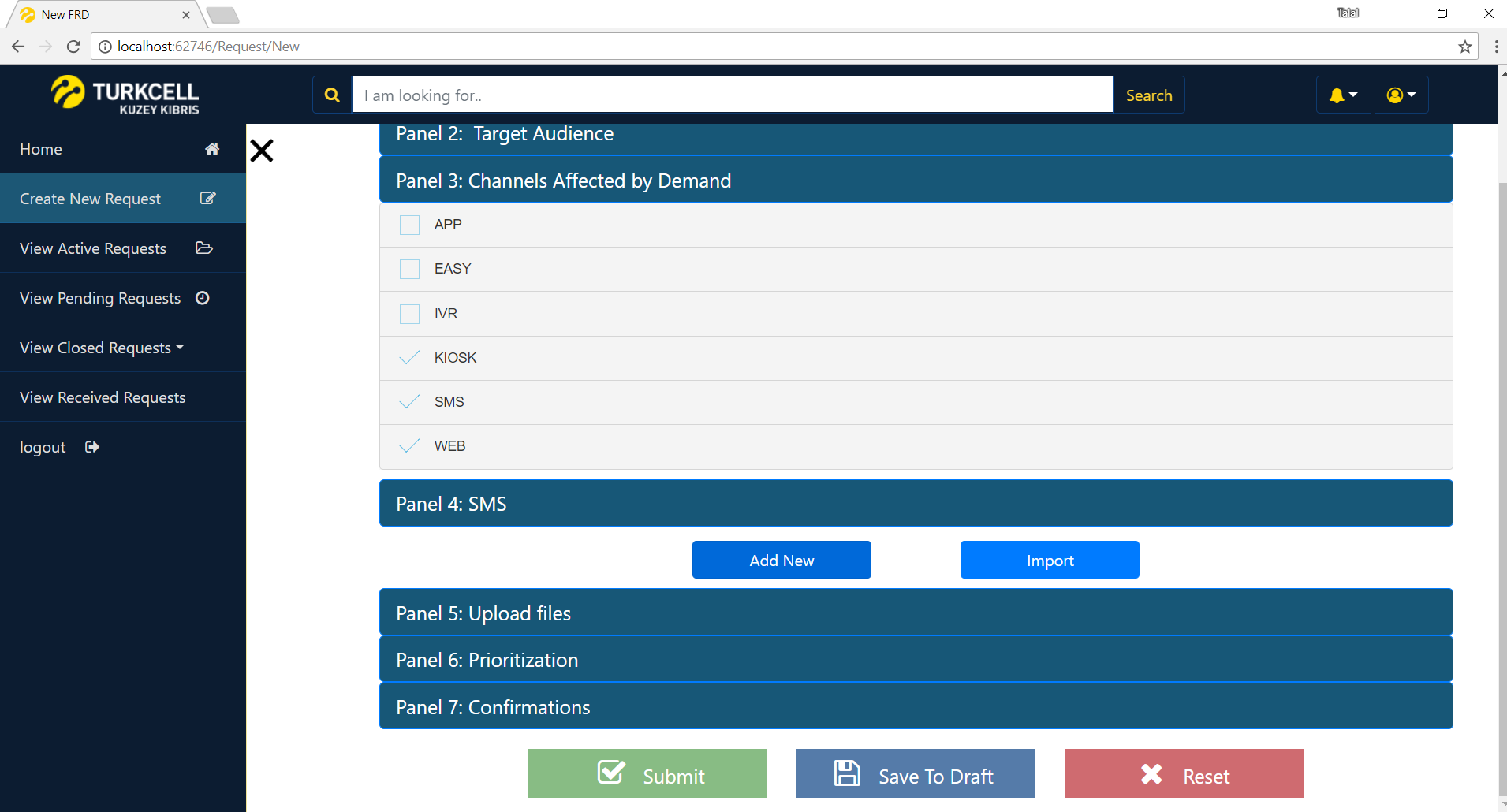


Fig. 40 Panel 3

# 

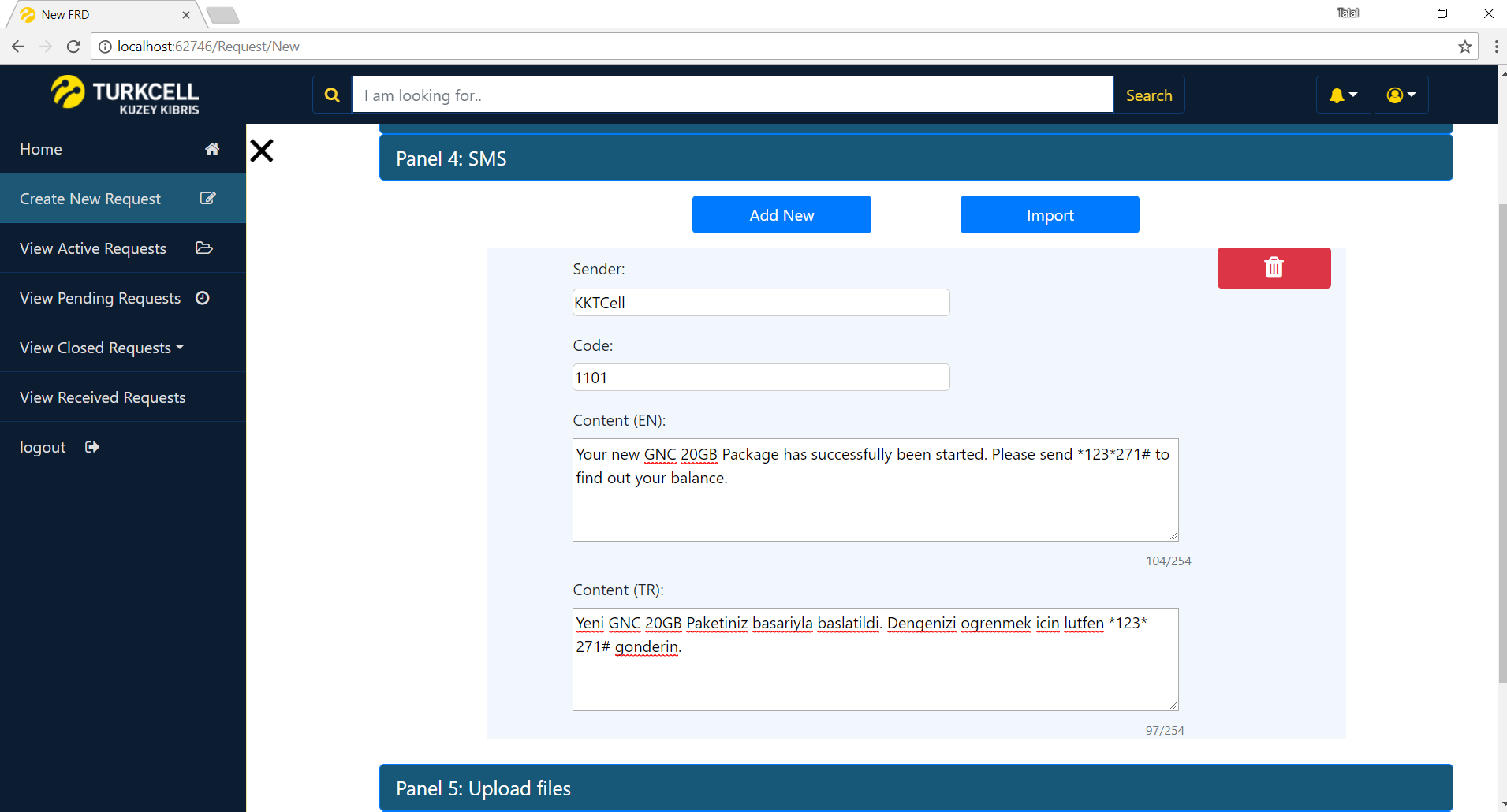


Fig. 41 Panel 4

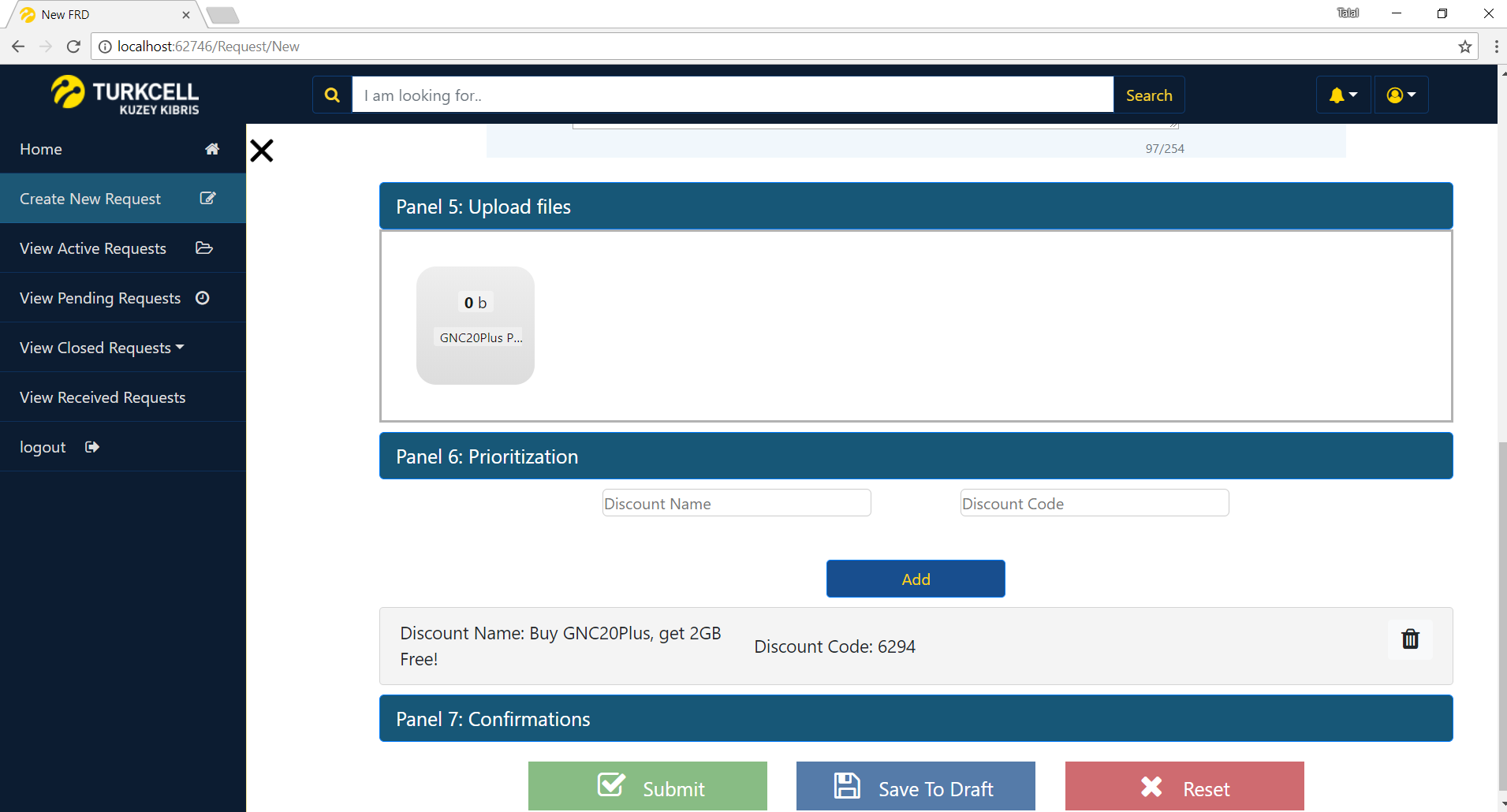


Fig. 42 Panel 5 and Panel 6

# 

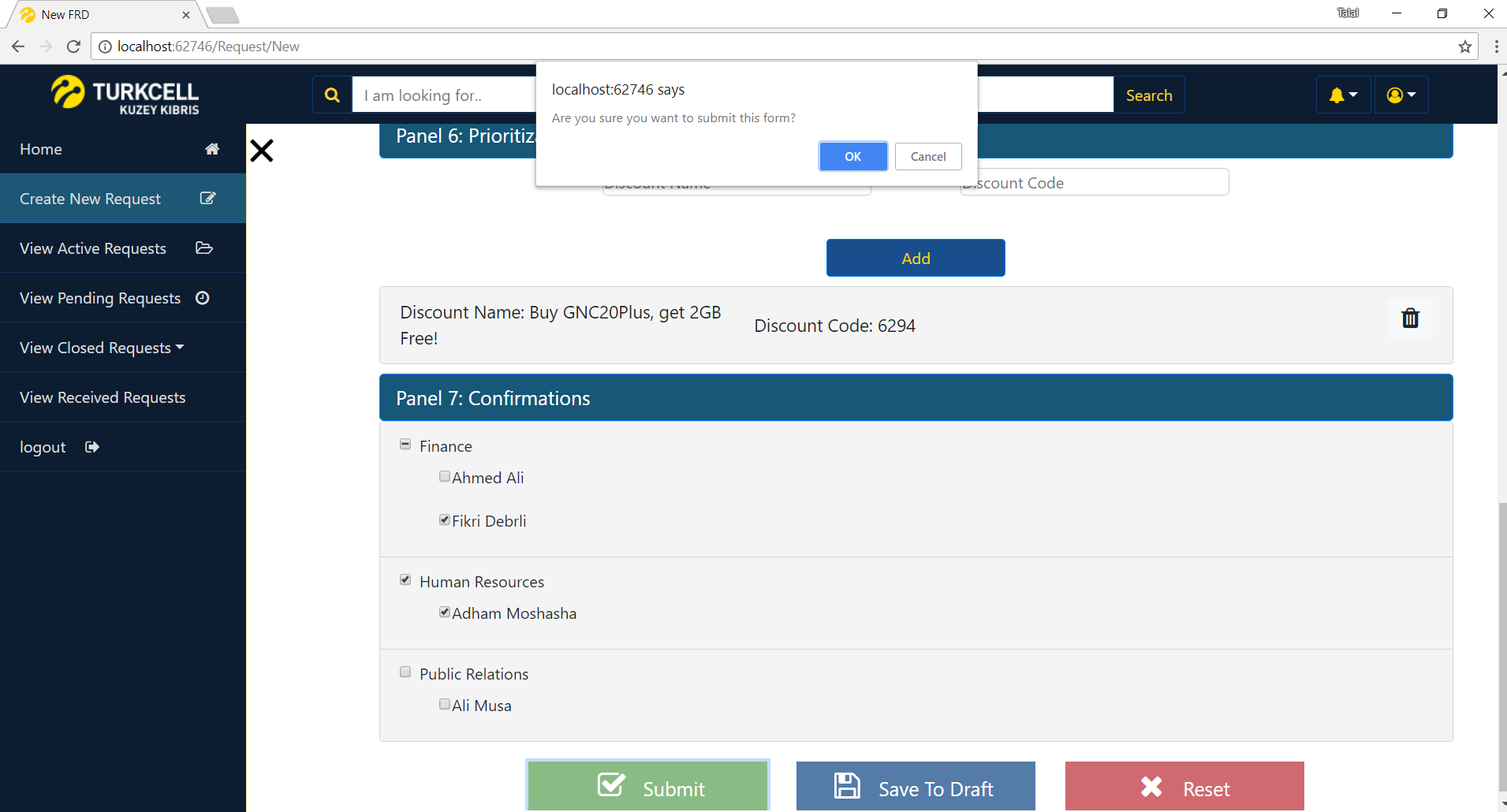


Fig. 43 Panel 7

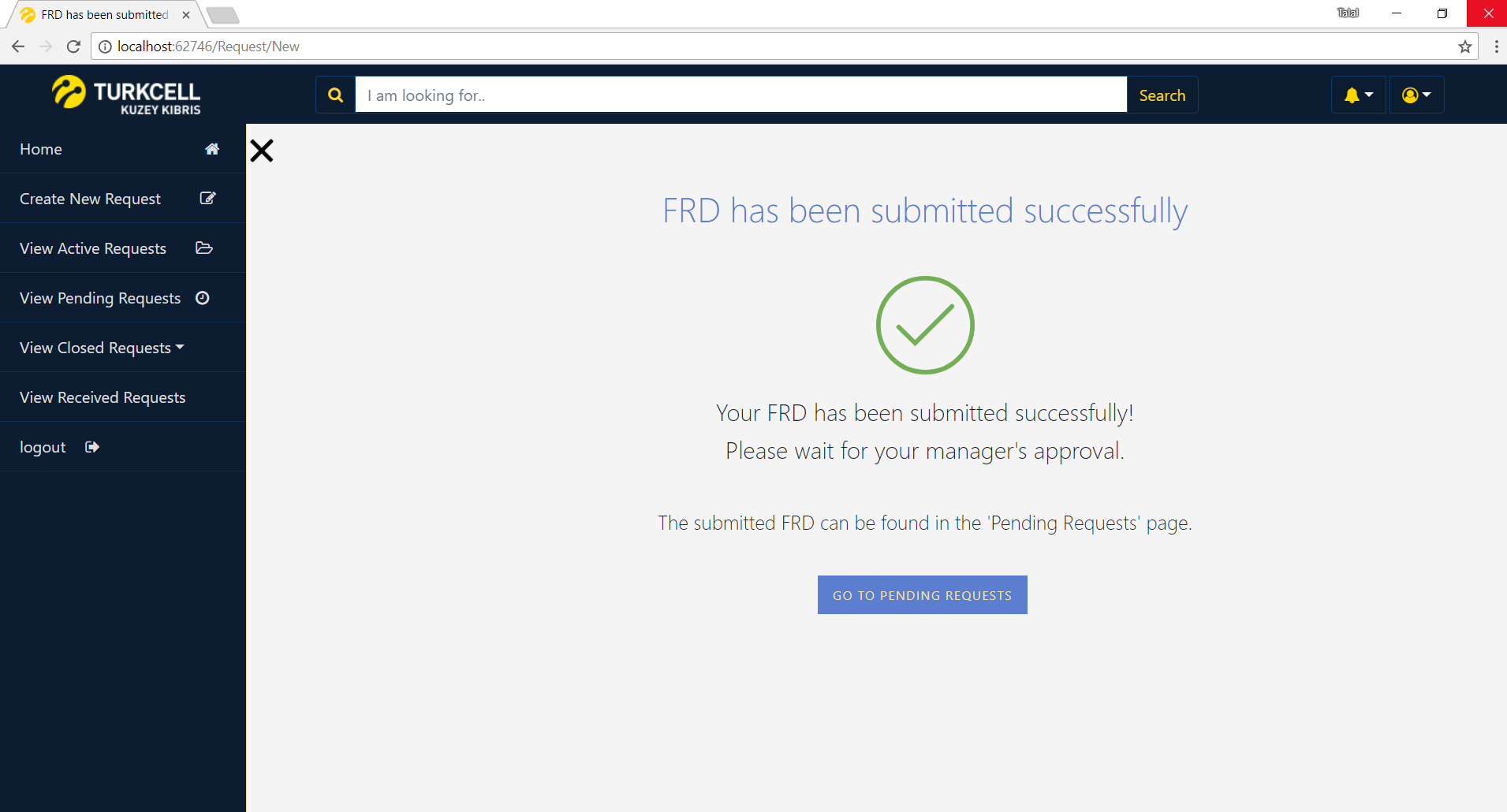


Fig. 44 FRD Submitted

# 

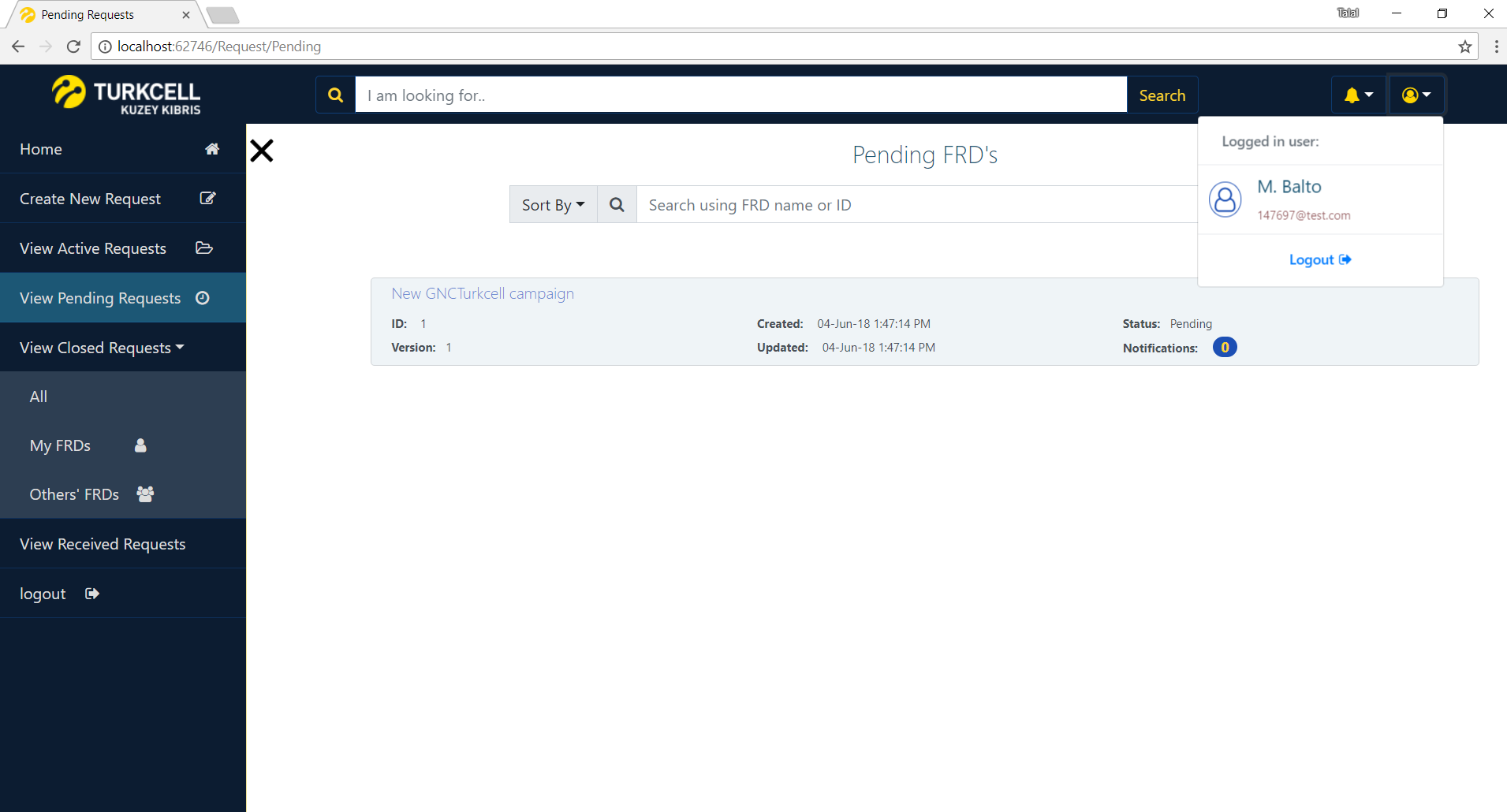


Fig. 45 FRD Status: Pending

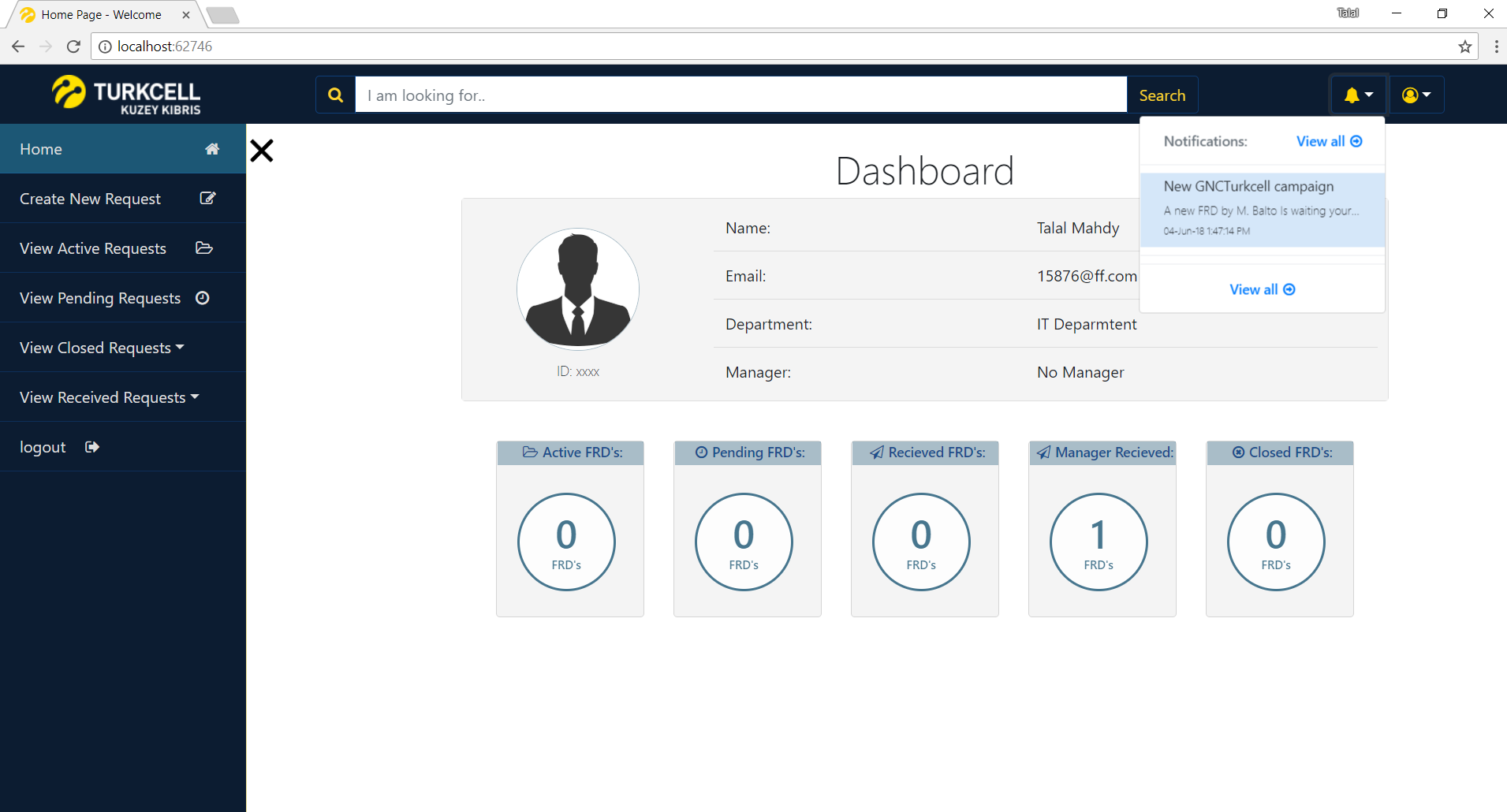


Fig. 46 Manager received FRD

# 

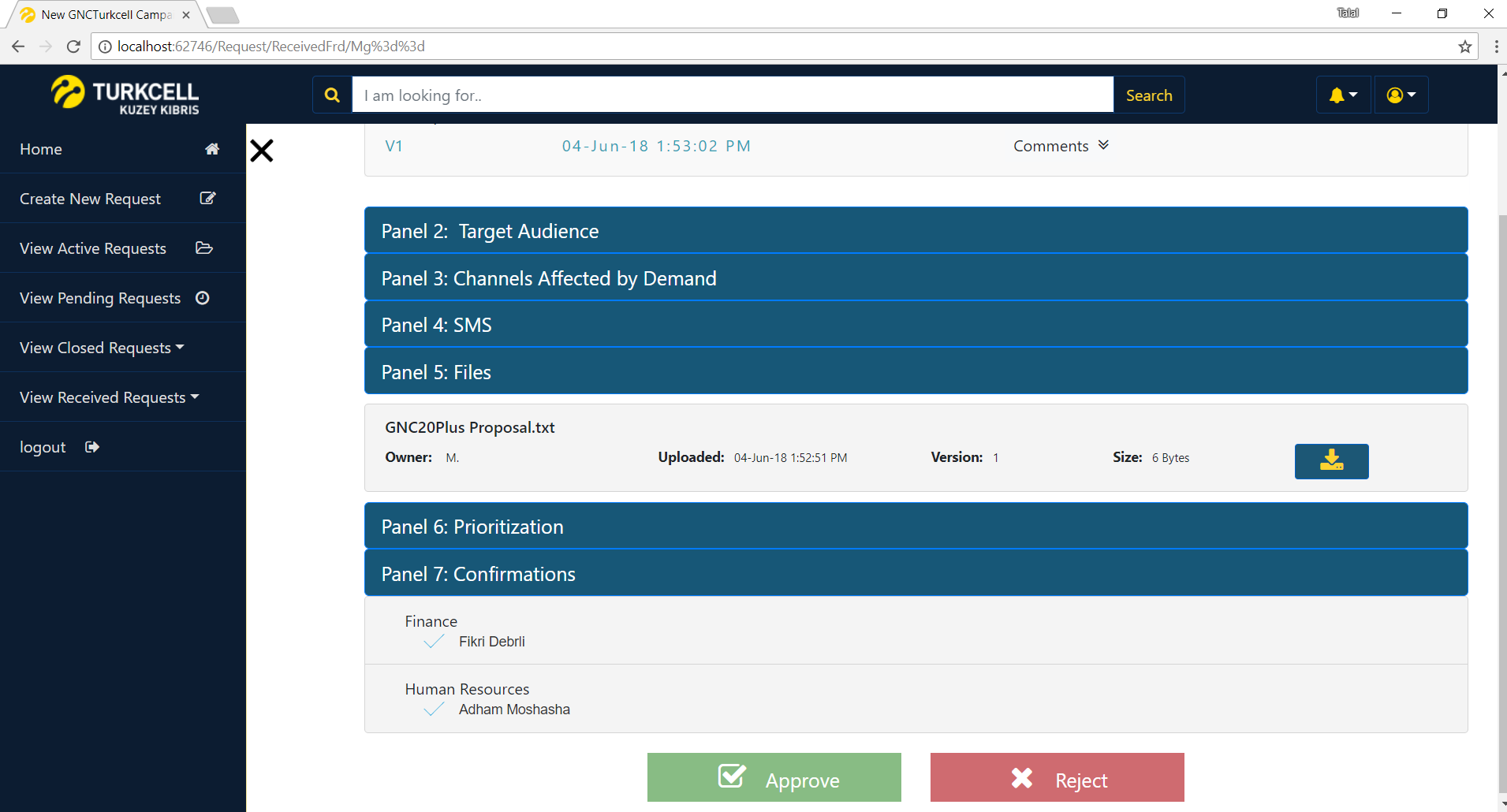


Fig. 47 Manager approves FRD

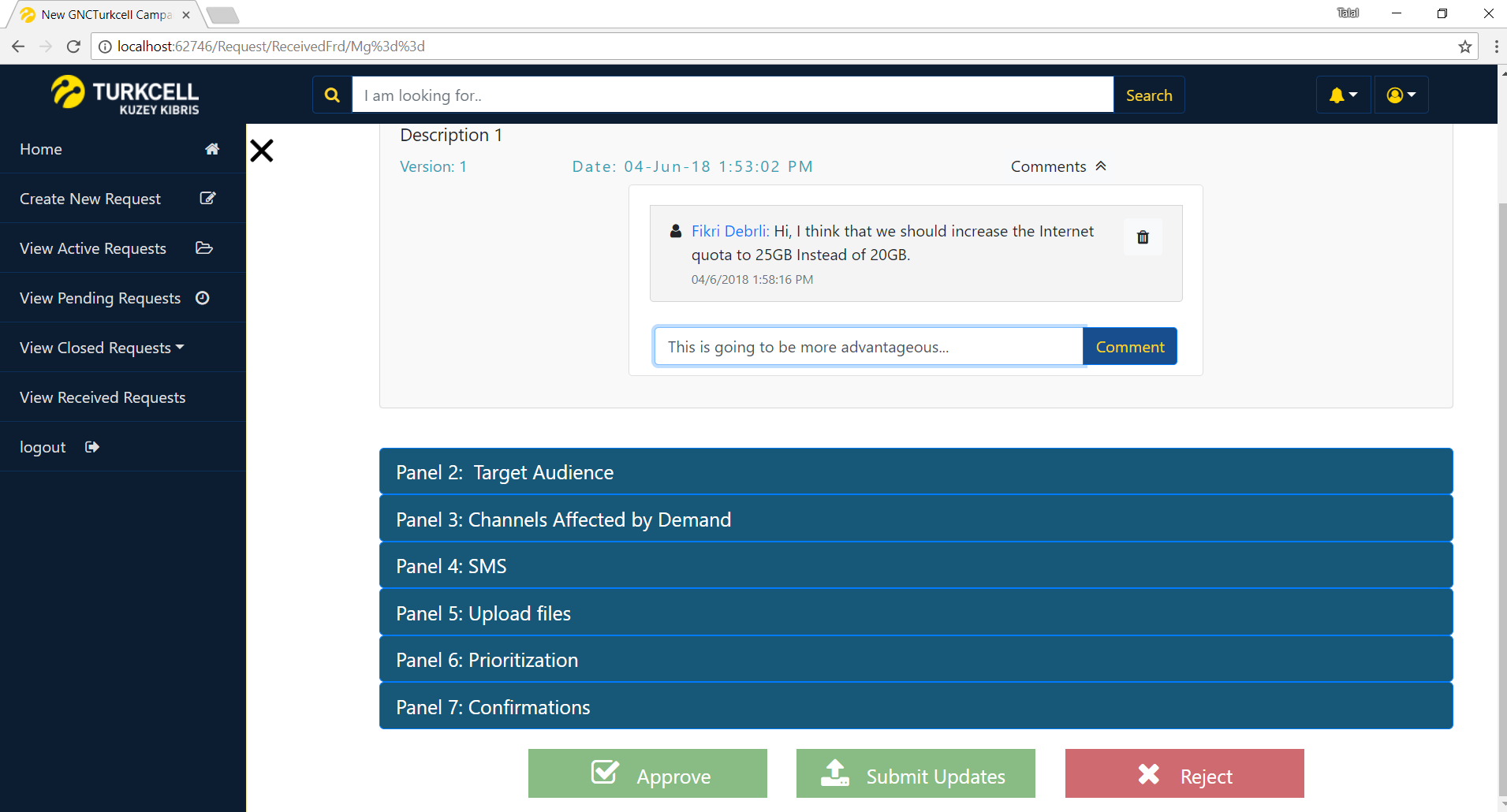


Fig. 48 User\_Fikri Collaborating after approval

# 8. DISCUSSION

The Change Request Management Application can be of great relief to all the Managers and Employees in the KKTCell Workplace. The Change Request Management Application is a not-for-profit and an open source project which can increase the performance of the employees and services at KKTCell Company. Furthermore, the Web Application for KKTCell can improve the organization and productivity in handling various requests, which in turn could improve the services at the National level in Northern Cyprus. The application’s abilities to handle even the most complicated of requests could prove crucial to the workflow of the employees, making them less worried about all the paperwork they had to fill out and making the various requests less stressful to file. Since a Client-Server Architecture will be implemented at the KKTCell Workplace, it will be very easy for employees to visit their coworker’s offices and cooperate on the request details. This project also acts as win-win situation for both the environment and the managers at KKTCell as it will greatly reduce the amount of papers required to be used daily and the managers may save on the costs of all the papers and use those savings to improve other services for the consumers. The Change Request Management Application request handling abilities could assist in the introduction of some bigger & more organized service or the introduction of a new technology in Northern Cyprus such as 4G LTE or 5G. The Change Request Management Application has been customized and developed based on the requirements provided by KKTCell and it is strongly recommended that other companies around the world follow the same approach in developing productive applications like this.

# 9. CONCLUSION

In this report, we covered the production of a Client-Server Web Application system known as the Change Request Management Application. First of all, we discussed the aims of the system which are to handle the various requests at the KKTCell workplace and to increase the productivity of its employees. We then discussed how the whole project was managed and organized, after which we discussed the various requirements and design aspects of the project. After the crucial requirements analysis and software design stages were over for the modules, the actual implementation and coding of the various modules began, after which the functions of the modules were tested for any bugs or defects. Working on this graduation project has been a positive learning and collaborative experience for all of the four team members involved. Our team had around 8 months to complete the project and through proper management, teamwork, collaboration and dedication, we were able to successfully complete this project to the best of our abilities. The Change Request Management Application has the potential to grow further and further as new features gets added in the future and we predict a bright and successful future for the application. Change Request Management Application can also act as an example to other organizations on how a properly designed system acts like and other organizations could benefit from the application’s designs and functionalities in producing their own applications. Software Engineering students may also learn a lot of things by referring to this application. The development of this application helped us personally by giving us chance to learn and work with various new tools, technologies and libraries such as ASP.NET MVC, Oracle DB, C# backend, JavaScript, React, Bootstrap, Slack along with many crucial software engineering activities. We are also planning to release many new features in the future such as a Manager’s Dashboard to enable the managers to easily add modify the FRD form and to gain more control over it. In the end, we are going to install this web application on all the PC’s at the KKTCell workplace to make it easier for everyone to use it.

# 10. REFERENCES

[1]. Kuzey Kibris Turkcell. Retrieved from http://www.kktcell.com/

[2]. IEEE Recommended Practice for Software Requirements Specifications. (1998). doi:   
 10.1109/ieeestd.1998.88286

[3]. IEEE Recommended Practice for Software Design Descriptions. (1998). doi:   
 10.1109/ieeestd.1998.88828

[4]. Project Management Software | Microsoft Project. Retrieved from   
 https://products.office.com/en/project/project-and-portfolio-management-software

[5]. Visual Studio Team Services. Retrieved from https://www.visualstudio.com/team-services/

[6]. Modelio Open Source - UML and BPMN free modeling tool. Retrieved from   
 https://www.modelio.org/

[7]. Wireframe.cc - minimal wireframing tool. Retrieved from https://wireframe.cc/

[8]. Flowchart Maker & Online Diagram Software. Retrieved from https://www.draw.io/

[9]. Bootstrap Studio. Retrieved from https://bootstrapstudio.io/

[10]. Database | Cloud Database | Oracle. Retrieved from   
 https://www.oracle.com/database/index.html

[11]. Visual Studio IDE, Code Editor, VSTS, & App Center. Retrieved from   
 https://www.visualstudio.com/

[12]. JSON. Retrieved from https://www.json.org/

[13]. Bootstrap. Retrieved from https://getbootstrap.com/

[14]. Sass: Syntactically Awesome Style Sheets. Retrieved from https://sass-lang.com/

[15]. jQuery. Retrieved from https://jquery.com/

[16]. React - A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/

[17]. Getting Started with LINQ in C#. (2015). Retrieved from https://docs.microsoft.com/en-  
 us/dotnet/csharp/programming-guide/concepts/linq/getting-started-with-linq

[18]. webpack. Retrieved from https://webpack.js.org/

[19]. Babel · The compiler for writing next generation JavaScript. Retrieved from   
 https://babeljs.io/

[20]. List.js. Retrieved from http://listjs.com/

[21]. SHA-2. Retrieved from https://en.wikipedia.org/wiki/SHA-2

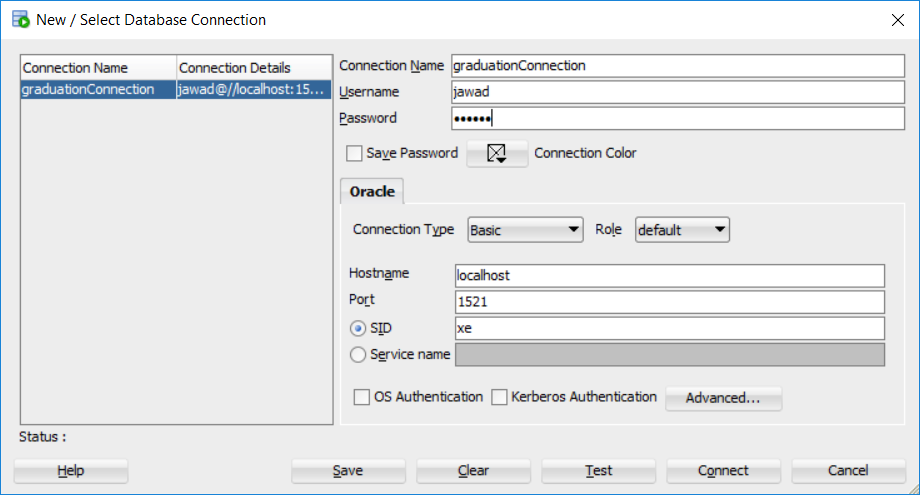
[22]. COCOMO. Retrieved from https://en.wikipedia.org/wiki/COCOMO

[23]. Ecma-262.pdf. (2017). Retrieved from http://www.ecma-  
 international.org/publications/files/ECMA-ST/Ecma-262.pdf

# APPENDIX A

**Instructions for Installing the system:**

1. Install Oracle Database if it is not already installed (Included in the CD).
2. Install Oracle SQLDeveloper if it is not already installed (Included in the CD).
3. Install LINQConnect if it is not already installed (Included in the CD).
4. Open SQLDeveloper.exe, create a new Database connection with   
   username: jawad,   
   password: 123456 as depicted below.



1. Open the The\_DB.sql file, copy all its contents (CTRL+A) and paste it in the SQLDeveloper worksheet textfield. Click Run Script (F5).
2. The database is now ready and you can now import the project using Visual Studio and run it.
3. The project is published on the CD with a file called FRD SYSTEM\_Published.rar.
4. The original project files of Visual Studio are also included.

# APPENDIX B

**Code for the system: 1LoginPage.xaml.cs**

This is an important C# code snippet for the application called RequestController.cs. The code for all the other parts of the program can be found on the CD attached with the report.

using System;

using System.Collections.Generic;

using System.Web.Mvc;

using turkcell\_web\_app.Models;

using turkcell\_web\_app.Models.Panels;

using turkcell\_web\_app.Models.Panels.Classes;

using turkcell\_web\_app.ViewModels;

using turkcell\_web\_app.Models.DB\_Adapters;

using turkcell\_web\_app.Models.Session;

using turkcell\_web\_app.Models.FRD\_Process;

using System.Runtime.Serialization.Formatters.Binary;

using System.IO;

using System.Text;

namespace turkcell\_web\_app.Controllers

{

[Authorize]

public class RequestController : Controller

{

// GET: Request

[HttpGet]

public ActionResult New(String message)

{

ViewBag.Message = message;

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

//check for saved FRD here

byte[] file = DB\_Functions.GetSaved();

if (file != null)

{

if (String.IsNullOrEmpty(message))

{

ViewBag.Message = "Previously saved FRD data loaded successfully. To erase, click on the RESET button";

}

Frd myfrd = G\_Functions.DeserializeFromBytes<Frd>(file);

myfrd.Type = Frd.TypesEnum.Saved;

return View(SavedFrdHandling.ProcessInfo(myfrd));

}

//get list of target audience from database and store in variable a

List<TargetAudience> a = DB\_Functions.DefaultTargets();

//get list of channels from database and store in variable b

List<Channel> b = DB\_Functions.DefaultChannels();

//get list of departments with their respective employees inside from database and store in

//variable c

List<Distribution\_Groups> c = DB\_Functions.DefaultGroups();

var senders = DB\_Functions.DefaultSmsSenders();

var viewModel = new Frd()

{

Type = Frd.TypesEnum.New,

Upload\_Token = DB\_Functions.NewUploadToken(DB\_Functions.UploadTokenType.New),

Panel0 = new Panel\_0()

{

Owner = Functions.GetName()

},

Panel2 = new Panel\_2()

{

Targets = a

},

Panel3 = new Panel\_3()

{

Channels = b

},

Panel4 = new Panel\_4()

{

Senders = senders

},

Panel7 = new Panel\_7()

{

DistributionGroups = c

}

};

return View(viewModel);

}

[ValidateAntiForgeryToken]

[HttpPost]

public ActionResult New(Frd submittedFrd)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

switch (submittedFrd.SubmitType)

{

case Frd.ButtonsEnum.Reset://reset

DB\_Functions.DeleteSavedDraft();

return RedirectToAction("New", new { message = "The FRD data has been reset successfully!" });

case Frd.ButtonsEnum.Submit://submit

if (DB\_Functions.SubmitNewFrd(submittedFrd))

{

SuccessData viewModel = new SuccessData()

{

Heading = "FRD has been submitted successfully",

PrimaryParagraph = "Your FRD has been submitted successfully!",

SecondaryParagraph = "Please wait for your manager's approval.",

LastParagraph = "The submitted FRD can be found in the 'Pending Requests' page.",

ButtonText = "GO TO PENDING REQUESTS",

ButtonLink = "Request/Pending"

};

return View("Success", viewModel);

}

else

{

submittedFrd.Type = Frd.TypesEnum.Saved;

return RedirectToAction("New", SavedFrdHandling.ProcessInfo(submittedFrd));

}

case Frd.ButtonsEnum.Save://save

DB\_Functions.SaveToDraft(G\_Functions.ToByteArray(submittedFrd));

return RedirectToAction("New", new { message = "The FRD data has been saved successfully!" });

}

var a = FRD\_Submit.Save(submittedFrd);

return RedirectToAction("New");

}

[HttpGet]

public ActionResult Active()

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

var a = DB\_Functions.ActiveList();

FrdsList viewModel = new FrdsList()

{

List = a

};

if (viewModel.List.Count > 0)

{

return View(viewModel);

}

else

{

NoFRD viewModel1 = new NoFRD()

{

Heading = "You currently have no active FRDS available",

PrimaryParagraph = "To create and submit a new FRD, go to 'New Request' page",

SecondaryParagraph = "",

LastParagraph = "",

ButtonText = "GO TO NEW REQUEST",

ButtonLink = "Request/New",

Page = NoFRD.menuitem.Active

};

return View("noFRD", viewModel1);

}

}

[HttpGet]

public ActionResult Pending()

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

var a = DB\_Functions.PendingList();

FrdsList viewModel = new FrdsList()

{

List = a

};

if (viewModel.List.Count > 0)

{

return View(viewModel);

}

else

{

NoFRD viewModel1 = new NoFRD()

{

Heading = "There are currently no pending FRDS available",

PrimaryParagraph = "FRDs appear as pending if you submitted an FRD and you are waiting for a manager's approval",

SecondaryParagraph = "To create and submit a new FRD, go to 'New Request' page",

LastParagraph = "",

ButtonText = "GO TO NEW REQUEST",

ButtonLink = "Request/New",

Page = NoFRD.menuitem.Pending

};

return View("noFRD", viewModel1);

}

}

[HttpGet]

public ActionResult Closed(string op = "all")

{

string frdMessage = "";

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

List<ActiveClosedFRDItem> list;

int op\_int = 0;

switch (op)

{

case "all":

op\_int = 0;

list = DB\_Functions.ClosedList(op\_int);

break;

case "me":

op\_int = 1;

list = DB\_Functions.ClosedList(op\_int);

break;

case "other":

op\_int = 2;

list = DB\_Functions.ClosedList(op\_int);

break;

default:

op\_int = 0;

list = DB\_Functions.ClosedList(op\_int);

break;

}

var viewModel = new FrdsList()

{

List = list

};

if (viewModel.List.Count > 0)

{

return View(viewModel);

}

else

{

switch (op)

{

case "all":

frdMessage = "";

break;

case "me":

frdMessage = "owned by you";

break;

case "other":

frdMessage = "owned by others";

break;

}

NoFRD viewModel1 = new NoFRD()

{

Heading = "There are currently no Closed FRDS " + frdMessage,

PrimaryParagraph = "FRDs "+ frdMessage + " are listed here if they have been closed",

SecondaryParagraph = "Your active FRDs can be found in the 'Active Requests' page",

LastParagraph = "",

ButtonText = "GO TO ACTIVE REQUESTS",

ButtonLink = "Request/Active",

Page = op\_int == 0 ? NoFRD.menuitem.ClosedAll : op\_int == 1 ? NoFRD.menuitem.ClosedMe : NoFRD.menuitem.ClosedOthers

};

return View("noFRD", viewModel1);

}

}

[HttpGet]

public ActionResult ReceivedU()

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

FrdsList viewModel = new FrdsList { List = DB\_Functions.ReceivedListAsUser() };

if (viewModel.List.Count > 0)

{

return View(viewModel);

}

else

{

NoFRD viewModel1 = new NoFRD()

{

Heading = "There are currently no received FRDS available",

PrimaryParagraph = "Received FRDs are listed here if you were selected as a recepient of the FRD during its creation by an employee",

SecondaryParagraph = "",

LastParagraph = "",

ButtonText = "GO TO HOME PAGE",

ButtonLink = "Home/Index",

Page = NoFRD.menuitem.ReceivedU

};

return View("noFRD", viewModel1);

}

}

[HttpGet]

public ActionResult ReceivedM()

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

if (!Functions.IsManager())

{

return new HttpNotFoundResult();

}

FrdsList viewModel = new FrdsList { List = DB\_Functions.ReceivedListAsManager() };

if (viewModel.List.Count > 0)

{

return View(viewModel);

}

else

{

NoFRD viewModel1 = new NoFRD()

{

Heading = "There are currently no received FRDS available",

PrimaryParagraph = "Received FRDs are listed here if you are a manager of an employee who submitted an FRD",

SecondaryParagraph = "",

LastParagraph = "",

ButtonText = "GO TO HOME PAGE",

ButtonLink = "Home/Index",

Page = NoFRD.menuitem.ReceivedM

};

return View("noFRD", viewModel1);

}

}

[HttpGet]

public ActionResult ActiveFrd(string Id,string error=null)

{

ViewBag.message = error;

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd File = DB\_Functions.CheckActive(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(Id)));

if (File == null)

{

//Error page

return RedirectToAction("Active");

}

Frd viewModel = ActiveFrdProcessor.Process(File);

viewModel.Upload\_Token = DB\_Functions.NewUploadToken(DB\_Functions.UploadTokenType.Active);

return View(viewModel);

}

[HttpGet]

public ActionResult PendingFrd(string Id)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd File = DB\_Functions.CheckPending(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(Id)));

if (File == null)

{

//Error page

return RedirectToAction("Active");

}

Frd viewModel = ActiveFrdProcessor.Process(File);

return View("View\_NoEdit", viewModel);

}

[HttpGet]

public ActionResult ReceivedFrd(string Id, string error= null)

{

ViewBag.message = error;

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd File = DB\_Functions.CheckReceived(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(Id)));

if (File == null)

{

//Error page

return RedirectToAction("Received");

}

Frd viewModel = ActiveFrdProcessor.Process(File);

//viewModel.Type = Frd.TypesEnum.ReceivedManager;

if (viewModel.Type == Frd.TypesEnum.ReceivedManagerPending || viewModel.Type == Frd.TypesEnum.ReceivedManagerActive)

{

return View("View\_NoEdit", viewModel);//Manager Approval

}

viewModel.Upload\_Token = DB\_Functions.NewUploadToken(DB\_Functions.UploadTokenType.Received);

return View(viewModel);

}

[HttpPost]

public ActionResult ReceivedFrd(Frd document)

{

SuccessData viewModel;

switch (document.SubmitType)

{

//reject, frd id is given back

case Frd.ButtonsEnum.Reject:

if (document.Type == Frd.TypesEnum.ReceivedManagerPending)

{

if (DB\_Functions.ManagerReject(document.Panel0.Id, document.VersionNotes))

{

viewModel = new SuccessData()

{

Heading = "FRD has been refuted successfully",

PrimaryParagraph = "the FRD has now been closed!",

SecondaryParagraph = "",

LastParagraph = "The rejected FRD can be found in the 'Closed Requests' page.",

ButtonText = "GO TO CLOSED REQUESTS",

ButtonLink = "Request/Closed"

};

return View("Success", viewModel);

}

else//if failed

{

}

}

else //if its others

{

if (DB\_Functions.User\_Reject(document.Panel0.Id))

{

viewModel = new SuccessData()

{

Heading = "FRD has been refuted successfully",

PrimaryParagraph = "",

SecondaryParagraph = "",

LastParagraph = "",

ButtonText = "GO TO HOME PAGE",

ButtonLink = "Home/Index"

};

return View("Success", viewModel);

}

}

break;

//approve, frd id is given back

case Frd.ButtonsEnum.Approve:

if (document.Type == Frd.TypesEnum.ReceivedManagerPending)

{

if (DB\_Functions.ManagerApprove(document.Panel0.Id))

{

viewModel = new SuccessData()

{

Heading = "FRD has been approved successfully",

PrimaryParagraph = "the active FRD submitted has been approved successfully!",

SecondaryParagraph = "The FRD has now been forwarded to the appropriate employees",

LastParagraph = "You can continue viewing this FRD on the 'RECEIVED REQUESTS' page",

ButtonText = "GO TO the RECEIVED REQUESTS page",

ButtonLink = "request/receivedM"

};

return View("Success", viewModel);

}

else

{

}

}

else

{

if (DB\_Functions.User\_Approve(document.Panel0.Id))

{

viewModel = new SuccessData()

{

Heading = "FRD has been approved successfully",

PrimaryParagraph = "the active FRD submitted has been approved successfully!",

SecondaryParagraph = "The FRD has now been forwarded to the appropriate employees",

LastParagraph = "You can continue viewing this FRD on the 'RECEIVED REQUESTS' page",

ButtonText = "GO TO the RECEIVED REQUESTS page",

ButtonLink = "request/receivedU"

};

return View("Success", viewModel);

}

else

{

}

}

break;

//submit UPDATES, frd id & list of comments are given back

case Frd.ButtonsEnum.Submit:

if (DB\_Functions.SubmitReceiverUpdate(document))

{

viewModel = new SuccessData()

{

Heading = "FRD updates have been submitted successfully",

PrimaryParagraph = "The active FRD has been updated successfully!",

SecondaryParagraph = "The appropriate employees have now received your updated version",

LastParagraph = "The updated FRD can still be found in the 'Received Requests' page.",

ButtonText = "GO TO RECEIVED REQUESTS",

ButtonLink = "Request/ReceivedU"

};

return View("Success", viewModel);

}

else

{

return RedirectToAction("ReceivedFrd", new { id = document.Panel0.Hashed\_Id, error = "Changes have to be made in order to submit an updated version of the FRD" });

}

default:

break;

}

return null;

}

[HttpPost]

public ActionResult ActiveFrd(Frd submittedForm)

{

SuccessData viewModel;

switch (submittedForm.SubmitType)

{

case Frd.ButtonsEnum.Submit:

if (DB\_Functions.SubmitOwnerUpdate(submittedForm))

{

viewModel = new SuccessData()

{

Heading = "FRD updates have been submitted successfully",

PrimaryParagraph = "Your active FRD has been updated successfully!",

SecondaryParagraph = "The selected employees have now received your updated version",

LastParagraph = "The updated FRD can be found in the 'Active Requests' page.",

ButtonText = "GO TO ACTIVE REQUESTS",

ButtonLink = "Request/Active"

};

return View("Success", viewModel);

}

else

return RedirectToAction("ActiveFrd",new {id = submittedForm.Panel0.Hashed\_Id, error = "Changes have to be made in order to submit an updated version of the FRD" });

case Frd.ButtonsEnum.Reject:

if (DB\_Functions.OwnerClose(submittedForm.Panel0.Id))

{

viewModel = new SuccessData()

{

Heading = "FRD has been closed successfully",

PrimaryParagraph = "Your active FRD has been closed successfully!",

SecondaryParagraph = "",

LastParagraph = "The closed FRD can be found in the 'Closed Requests' page.",

ButtonText = "GO TO Closed REQUESTS",

ButtonLink = "Request/Closed"

};

return View("Success", viewModel);

}

else

{

}

break;

}

return View();

}

[System.Web.Mvc.Route("ViewFrd/{V}/{Id}")]

[HttpGet]

public ActionResult ViewFrd(string Id, int V)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd File = DB\_Functions.CheckForFrd(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(Id)));

if (File == null)

{

//Error page

return new HttpNotFoundResult();

}

if (V != 0)

{

File.LatestVersion = V;

File.Panel0.LatestVersion = V;

}

Frd s = ActiveFrdProcessor.Process(File, V);

s.Type = Frd.TypesEnum.Closed;

return View("View\_NoEdit", s);

}

[HttpGet]

public ActionResult ClosedFrd(string Id)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd File = DB\_Functions.CheckClosed(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(Id)));

if (File == null)

{

//Error page

return RedirectToAction("Closed");

}

Frd s = ActiveFrdProcessor.Process(File);

return View("View\_NoEdit", s);

}

[HttpGet]

public ActionResult Search(string input)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

SearchResult viewModel = new SearchResult

{

Searched = input,

List = DB\_Functions.Search(input)

};

return View(viewModel);

}

[HttpGet]

public ActionResult SearchedFrd(string FrdId,int Version=0)

{

if (Functions.NoSession())

{

return RedirectToAction("Login", "Authentication");

}

Frd frd = DB\_Functions.CheckForFrd(FrdId);

if (frd == null)

return new HttpNotFoundResult();

if (Version==0 || frd.LatestVersion == Version)

{

switch (frd.Type)

{

case Frd.TypesEnum.Active:

return RedirectToAction("ActiveFrd", "Request", new { Id = G\_Functions.GetHash(FrdId)});

case Frd.TypesEnum.Closed:

return RedirectToAction("ClosedFrd", "Request", new { Id = G\_Functions.GetHash(FrdId) });

break;

case Frd.TypesEnum.Pending:

return RedirectToAction("PendingFrd", "Request", new { Id = G\_Functions.GetHash(FrdId) });

break;

case Frd.TypesEnum.ReceivedManagerActive:

case Frd.TypesEnum.ReceivedManagerPending:

case Frd.TypesEnum.ReceivedUser:

return RedirectToAction("ReceivedFrd", "Request", new { Id = G\_Functions.GetHash(FrdId) });

break;

}

}

else

{

return RedirectToAction("ViewFrd", "Request", new { Id = G\_Functions.GetHash(FrdId), V = Version });

}

return new HttpNotFoundResult();

}

}

}

# APPENDIX C

**Abbreviations and Definitions:**

API: Application Program Interface

API: Application Programming Interface

ASP: Active Server Pages

BPMN: Business Process Model Notation

BPMN: Business Process Modeling Notation

DB: Database

EMU: Eastern Mediterranean University

ERD: Entity Relationship Diagram

FP: Function Points

FRD: Function Requirements Document

GUI: Graphical User Interface

IDE: Integrated Development Environment

IEEE: Institute of Electrical and Electronics Engineers

JSON: JavaScript Object Notation

KKTCell: Kuzey Kibris Turkcell

LINQ: Language Integrated Query

MAU: Monthly Average Users

OS: Operating System

R&D: Research and Development

SDS: Software Design Specification Document

SQL: Structured Query Language

SRS: Software Requirements Specification Document

Sass: Syntactically Awesome Style Sheets

TCF: Technical Complexity Factor

TUBITAK: The Scientific and Technological Research Council of Turkey

UFP: Unadjusted Function Points

UI: User Interface

UML: Unified Modeling Language