JVMBEK Project Management System

Master Test Plan

Team JVMBEK

1 April 2015

Prepared by:

Name	ID Number
Karim Atwa	27716273
Edward Chiang	26676383
Mark Le	26991203
Benjamin Slapcoff	26663850
Vinh Truong	26941141
Chi Kit Tsang	25692636

Prepared for:
COMP 354 - Introduction to Software Engineering
Instructor: Dr. Greg Butler

Contents

1 Introduction

The primary goal of this project is to develop a program to manage projects and tasks. Using this program, managers can obtain critical information about various projects, and can assign members to specified tasks. The program can also generate Gantt charts and perform critical path, PERT, and earned value analyses for a given project.

2 Test Plan

This document provides a plan to test the proper functionality of the JVMBEK Project Management System, developed by our group. It contains database integrity, unit, functional, integration and user interface testing. These different types of testing are ordered from most to least essential; later types of testing require that earlier types of testing be successful. For instance, the database integrity must be assuredly sound before elements can be added to or removed from it, as occurs in unit testing. As well, test cases have been elaborated to accompany the various types of testing. As a general rule, test cases were designed to cover potential scenarios that the program would be likely to encounter during operation.

2.1 Data and Database Integrity Testing

This type of testing involves checking the quality of data in the databases in order to verify that the data in the database is accurate and functions as expected within a given application. We need to make sure that our database is indeed functioning as required in order to proceed with other tests that need access to the database.

The JVMBEK Project Manager software uses SQLite in order to manage the data at a local level. The data is stored and retrieved by the software when specific functions are called, and any user can only access data related to their account.

The data is stored in the form of tables:

- User Credentials: user ID (or username), password, user type, first and last names.
- Projects: project ID, project name, description, date created and start date
- Tasks: task ID, task name, duration, description, comment, progress, PERT optimistic time, PERT pessimistic time, PERT estimated time, PERT variance, start date and end date.

Their relationship in terms of parent and child to each other are also stored in tables:

- Manager-Project: Shows which project (child) belongs to which Project Manager (parent)
- Project-Tasks: Shows which task (child) belongs to which project (parent)
- Task-Members: Shows which member (child) is assigned to which task (parent).

Triggers are used to ensure that if a parent is removed from a table their children are also removed from the database in order to keep the size of the database down and help improve the performance of our software.

Essential pre-condition for all test cases: A project manager type user is used to connect to the database for the tests, and all test cases are to be done under the same user in order to retain consistency. The test will be run several times using different project manager users.

Test Case 1	Project creation
Pre-condition	No additional pre-conditions
Test Case Description	To ensure a created project is actually added to the database by adding a project and retrieving it to check if all fields are present.
Input	New project information (ID, name, description, date created, start date)
Expected Result	Returned project should contain the same data as the input project.
Obtained Result	Passed.

Test Case 2	Project-Owner relation
Pre-condition	A project has been created and is added the database.
Test Case Description	To ensure a created project belongs to the project manager who created it.
Input	None. The project has already been created and added to the database.
Expected Result	Returned task should contain the same data as the input task.
Obtained Result	Passed.

Test Case 3	Task creation
Pre-condition	A project has been created and is added the database.
Test Case Description	To ensure that a created task is actually added to the database by adding and retrieving a task to check if all fields are present.
Input	New task information (ID, name, duration, start and end dates)
Expected Result	Returned task should contain the same data as the input task.
Obtained Result	Passed.

Test Case 4	Project-Task relation
Pre-condition	A task has been created and is added the database.
Test Case Description	To ensure that that the task has been saved as a child of the project it was created for.
Input	None.
Expected Result	Returned relation should contain the parent project ID and the ID of the task that was created.
Obtained Result	Passed.

Test Case 5	Task sequence relation
Pre-condition	One or more tasks have been created and added the database.
Test Case Description	To ensure that that the tasks sequence has been saved correctly when it was created. All task should have a preceding task inside the database. Starting tasks should have themselves as preceding task.
Input	None. The task has already been created and added to the database.
Expected Result	Returned relation should contain the current task ID and the ID of the preceding task.
Obtained Result	Passed.

Test Case 6	Assign member
Pre-condition	One or more tasks have been created and added the database.
Test Case Description	To ensure that the member assigned to a task actually exists within the database and the relation between the member and the task is indeed present.
Input	A selected task.
Expected Result	Returned relation should contain the assigned member's ID and the assigned task's ID.
Obtained Result	Passed.

Test Case 7	Task deletion
Pre-condition	One or more tasks have been created and added the database.
Test Case Description	To ensure a deleted task is actually removed from the database.
Input	A selected task.
Expected Result	The task should not exist in the database anymore and cannot be retrieved.
Obtained Result	Passed.

Test Case 8	Task relations deletion
Pre-condition	One or more tasks have been created and added the database and subsequently deleted from it.
Test Case Description	To ensure that the deletion of a task removed its associated relations (parent project relation, assigned members relation and preceding tasks relation).
Input	A selected task with other task associations.
Expected Result	The database shouldn't contain any relations associated to the deleted task any longer.
Obtained Result	Passed.

Test Case 9	Project deletion
Pre-condition	A project has been created and is added the database.
Test Case Description	To ensure a deleted project is actually removed from the database.
Input	A selected project.
Expected Result	The project should not exist in the database any longer and cannot be retrieved.
Obtained Result	Passed.

Test Case 10	Project relations deletion
Pre-condition	Test cases "Task deletion" and "Task relations deletion" must have both passed. A project has been created and added the database and subsequently deleted from it.
Test Case Description	To ensure that the deletion of a project removed the project-owner relation and all child tasks their associated relations.
Input	A selected project.
Expected Result	The database shouldn't contain any relations associated to the deleted project any longer.
Obtained Result	Passed.

2.2 Unit Testing

Unit testing is performed on individual units of code in order to find out whether they are working or not. In our case, we found it logical to consider the base unit to test as a program class. Therefore, each unit consists of a class, and tests are performed on some of the functions contained in each class, independently of each other. Testing techniques used include black box testing.

The following lists mention the functions that were tested from the various classes:
Class ProjectManager

- Function login

- Function getProjects
- Function setSelectedProject

Class Project

- Function getAssignedTasks
- Function getTaskById
- Function assignMember

Class Task

- Function getProject
- Function getPrecedingIds
- Function getPrecedingIdsAsString
- Function updateTaskSequence

2.2.1 Unit 1: ProjectManager Class

2.2.1.1 Function login

ProjectManager function login allows the user to log in using valid credentials. Therefore, it takes two strings as arguments, the first being a username and the second the corresponding password. Upon a successful login, the function will return a User object containing information about the user who just logged in.

Black Box Testing
Each test case will consider all possible input combinations that could be passed to the function via the user, including cases in which the user enters invalid data.

Test Case 1: Pass invalid username and password arguments
Test Case 2: Pass a valid username argument, but an invalid password argument
Test Case 3: Pass an invalid username argument, but a valid password argument
Test Case 4: Pass valid, but non-matching username and password arguments
Test Case 5: Pass both valid and matching username and password arguments

The expected result is a User object representing the user if the login was successful, null if not.

Obtained Results:
Test Cases 1 to 4: An error dialog appears, with the message ”The username or password you entered was invalid. Please try again.”
Test Case 5: The system prints out that the user has logged in either as a manager or member, depending on which is the case

2.2.1.2 Function getProjects

ProjectManager function getProjects takes no argument and should return all the projects under a ProjectManager as a HashMap. All managers can create projects and once a project has been created, it is saved to the database and can only be accessed by its owner, the project manager.

Black Box Testing
Each test case will be tested with a different Manager object. A given project manager can have no fewer than 0 project and there are no upper limit to the amount of projects a project manager can have, so the testing will cover a project manager having between 0 and 10 projects.

Test Case 1: The manager has 0 project
Test Case 2: The manager has 1 project
Test Case 3: The manager has 10 projects

The expected result is a HashMap containing the projects associated with the manager being returned.

Obtained Results:
Test Case 1: An empty HashMap is returned
Test Case 2: A HashMap containing the single project is returned
Test Case 3: A HashMap containing all 10 projects is returned

2.2.1.3 Function setSelectedProject

ProjectManager function setSelectedProject takes an integer value, the project ID, and should find a project belonging associated to the ProjectManager with the given project ID. Then the ProjectManager object’s currently selected project should be set to the project that was found.

Black Box Testing
Test cases will use different integer values. The test will cover values equal to the Project-

Manager object's projects' ID's and non-existing project ID's. Valid project ID's range from 0 to infinite.

Test Case 1: Pass an argument that is under the lower bound for the project ID range (value < 0)
Test Case 2: Pass an argument that is within the bounds for the project ID range, but that does not correspond to the ID of any existing project
Test Case 3: Pass an argument that is within the bounds for the project ID range and corresponds to the ID of an existing project

The expected result is that the currently selected project for the manager will be set as being the project with the passed ID, if it is valid. Otherwise, an error message should be displayed.

Obtained Results:
Test Cases 1 and 2: The system prints out an error message stating that the specified project could not be found
Test Case 3: The specified project becomes selected

2.2.2 Unit 2: Project Class

2.2.2.1 Function getAssignedTasks

Project function getAssignedTasks takes as argument an integer which represents a users ID and returns an ArrayList of tasks that have been assigned to that user.

Black Box Testing
The two test cases represent the two possible scenarios when a user ID is passed - valid or not.

Test Case 1: Call the function with an invalid user ID
Test Case 2: Call the function with a valid user ID

The expected result is an ArrayList containing all the tasks that have been assigned to the user with the given user ID. The ArrayList should be empty if there are no such tasks.

Obtained Results:
Test Case 1: An empty ArrayList is returned
Test Case 2: An ArrayList containing the tasks assigned to the specified user is returned

2.2.2.2 Function getTaskById

Project function getTaskById takes an integer as argument and returns a task with the same task ID.

Black Box Testing
The two test cases represent the two possible scenarios when a task ID is passed - valid or not.

Test Case 1: Call the function with an invalid task ID
Test Case 2: Call the function with a valid task ID

The expected result is a Task object with the corresponding task ID, null if such a task does not exist.

Obtained Results:
Test Case 1: A null Task object is returned
Test Case 2: The proper Task object is returned

2.2.2.3 Function assignMember

Project function assignMember takes as argument a User object and an integer representing a task ID, and assigns the specified user to the task with the given task ID. It does not explicitly return anything.

Black Box Testing
The two test cases represent the four possible combinations when passing a User

Test Case 1: Call the function with dummy (non-existing) User object and task ID
Test Case 2: Call the function with dummy User object, but valid task ID
Test Case 3: Call the function with valid User object, but dummy task ID
Test Case 4: Call the function with valid User object and task ID, but with the user already assigned to the task
Test Case 5: Call the function with valid User object and task ID, with the user not already assigned to the task

The expected result is that the user that has the given user ID, if a member, is assigned to the task that has the given task ID.

Obtained Results:

Test Cases 1 to 3: No assignment occurs
Test Case 4: No assignment occurs, and the system prints out an error message stating this
Test Case 5: The member is assigned to the task

2.2.3 Unit 3: Task Class

2.2.3.1 Function getProject

Task function getProject() takes no argument and should return the project to which the calling Task object belongs.

Black Box Testing
The function takes no argument, and so there is no variation on inputs that can be given.

The expected result is, as stated, the project to which the calling Task object belongs.

Obtained Result: Passed.

2.2.3.2 Function getPrecedingIds

Task function getPrecedingIds takes no argument and returns an ArrayList which should contain the Task objects ID's which precede the Task object that calls this function.

Black Box Testing
The function takes no argument, and so there is no variation on inputs that can be given.

The expected result is, as stated, an ArrayList containing all the Task objects ID's which precede the calling Task object.

Obtained Result: Passed.

2.2.3.3 Function getPrecedingIdsAsString

Task function getPrecedingIdsAsString takes no argument and returns as a string the ID's of the preceding task objects.

Black Box Testing

The function takes no argument, and so there is no variation on inputs that can be given.

The expected result is, as stated, the ID's of the Task objects preceding the calling Task object, but as a single string.

Obtained Result: Passed.

2.3 Functional Testing

Functional testing is a type of black box testing that involves checking whether or not the system can perform certain implemented functions properly, as per design requirements and specifications.

Since this system is designed to support project management, the vast majority of implemented functionalities were meant to be used only by managers, as per the specifications. Therefore, most functional testing will revolve around manager-specific functions.

The functions that were tested are as follows:
Manager-Specific Functions:

- Login as a Manager
- Create a Project
- Load a Project
- Assign a Member to a Project
- Modify a Project
- Delete a Project
- Create a Task
- Delete a Task
- Generate a Gantt Chart

Member-Specific Functions:

- Login as a Member
- View Assigned Tasks

2.3.1 Scope 1: Manager-Specific Functions

Manager-specific functions are those functions which can only be performed while the program is being used by a manager. This section comprises the majority of tested functions, due to the nature of the system.

2.3.1.1 Login as a Manager

Test Case	Login as a manager
Test Case Description	To ensure that entering Project Manager credentials will enable the user to log in a a manager.
Input	• Enter a manager's ID and corresponding password • Click on the button "Log in"
Output	The welcome screen is displayed, with the manager's name.

2.3.1.2 Create a Project

Test Case	Create a project with valid input
Test Case Description	To ensure that a project is actually created and inserted into the database when the "Create Project" button is clicked.
Input	• Log in as a manager • Go to project creation screen • Fill out the project name and set a start date using the correct format(DD-MM-YYYY) • Optionally, add a description to the project • Click on the button "Create Project"
Output	A project with the given information is generated and put in the database. It can be found in the drop-down menu of the Load Project screen.

Test Case	Create a project with invalid input
Test Case Description	To ensure that no project is created when invalid input is given by the user.
Input	• Log in as a manager • Go to project creation screen • Fill out the project name with a string that begins with a digit or symbol, or set a start date using an incorrect format (i.e. differing from the format DD-MM-YYYY) - or leave either one of these fields empty • Click on the button "Create Project"
Output	An error message will pop up depending on the error: • If the project name does not start with a letter, an error message will appear, stating that the naming is invalid • If the date entered is out of range or does not follow the correct format (DD-MM-YYYY), an error message will appear, stating that the given date is invalid • If at least one of the two fields was left empty, an error message will appear, notifying the user of this

2.3.1.3 Load a Project

Test Case	Select and load a project
Test Case Description	To ensure that a project can be selected and loaded properly from the database.
Input	• Log in as a manager • Go to project selection screen • Select a project from the drop-down box • Click on the button ”Load”
Output	The selected project is loaded and its information is shown in the program. The user is brought to the Project Tasks screen, and can view tasks associated with this project.

2.3.1.4 Assign a Member to a Task

Test Case	Assign a member to a task
Test Case Description	To ensure that the selected member is actually assigned to a selected task.
Input	• Log in as a manager • Load a project • Go to the project members screen by clicking on the ”Show Members” button • Click on the ”Assign Member” button on the screen • Enter a task ID, and select a member using the drop-down box • Click on the button ”Assign”
Output	The member is assigned the specified task; this should be reflected in the database.

2.3.1.5 Modify a Project

Test Case	Modify a project’s information
Test Case Description	To ensure that the selected project’s information is updated in the database.
Input	• Log in as a manager • Go to the project selection screen by clicking on the ”Load” button • Select the targeted project using the drop-down box • Click on the ”Modify” button on the screen • Change the project’s name and description as wished • Click on the button ”Update Project”
Output	The project’s name and description should be updated accordingly in the database, unless the given project name is invalid. In this case, an error message will be displayed.

2.3.1.6 Delete a Project

Test Case	Delete a project
Test Case Description	To ensure a project can be properly deleted from the database
Input	• Log in as a manager • Go to the project selection screen by clicking on the ”Load” button • Select the targeted project using the drop-down box • Click on the ”Delete” button on the screen • Confirm the deletion by clicking on the ”Yes” button in the pop-up dialog box
Output	The project’s name and description should be removed accordingly from the database, and it should not appear in the drop-down project selection box any longer.

2.3.1.7 Create a Task

Test Case	Create a task with valid input
Test Case Description	To ensure that a task can be added to a project correctly.
Input	• Log in as a manager • Go to the task creation screen by clicking on the "Add Task" button • Fill out the required fields, meeting the following requirements: – Start and End Dates: Must follow the DD-MM-YYYY format – Optimistic: Must be an integer equal to or greater than 0, and smaller than the duration of the task (end date - start date). – Pessimistic: Must be an integer equal to or greater than 0, and larger than the duration of the task (end date - start date). – Preceding Tasks: Either check the box "No Preceding Task", or leave it unchecked and select at least one preceding task • Click on the "Create Task" button on the screen
Output	A new task is created and the user is brought to the Project Tasks screen. The created task should appear on the screen. Its default status should be "In Queue".

Test Case	Create a task with invalid input
Test Case Description	To ensure that a task with invalid input will not be created.
Input	• Log in as a manager • Go to the task creation screen by clicking on the "Add Task" button • Fill out the required fields, meeting at least one of the following requirements: – Enter an incorrect task name by starting with something that is not a letter or leave it empty – Enter an incorrect start date by not using the correct format or entering a date that does not exist, or leave it empty. – Enter an incorrect end date by not using the correct format or entering a date that does not exist or a date on or before the start date, or leave it empty. – Entering an optimistic time that is negative or larger than the duration (end date start date) or leave it empty. – Entering a pessimistic time that is negative or smaller than the duration (end date start date) or leave it empty. – Set no preceding task by not leaving unchecked the "No Preceding Task" checkbox and not selecting a preceding task. • Click on the "Create Task" button on the screen
Output	An error message will pop up depending on the error: • If the project name is empty does not begin with a letter, the "Incorrect Naming" error message will appear. • If the start date entered is out of bound or does not follow the correct format (DD-MM-YYYY) the "Incorrect Date Format" error will appear. • If the end date entered is out of range, does not follow the correct format (DD-MM-YYYY) or is smaller than the start date, the "Invalid Date Entry" error will pop up. • If the optimistic time field is empty, negative or larger than the duration, an error message stating this will appear. • If the pessimistic time field is empty, negative or smaller than the duration, an error message stating this will appear.

2.3.1.8 Delete a Task

Test Case	Delete a task
Test Case Description	To ensure that a task can be deleted from a project correctly.
Input	• Log in as a manager • Load a project that has tasks already • Select a task in the table • Click on the ”Delete Task” button on the screen • Confirm the deletion by clicking on the ”Yes” button in the pop-up dialog box
Output	A task is deleted from the database, and does not appear in the table any longer.

Test Case	Delete a task incorrectly
Test Case Description	To ensure that the program does not update the database when no task is selected for deletion.
Input	• Log in as a manager • Load a project • Click on the ”Delete Task” button on the screen without selecting a task
Output	A ”No Task Selected” error should appear.

Test Case	Cancel a task deletion
Test Case Description	To ensure that a task deletion can be cancelled properly.
Input	• Log in as a manager • Load a project that has tasks already • Select a task in the table • Click on the "Delete Task" button on the screen • Cancel the deletion by clicking on the "No" button in the pop-up dialog box
Output	Nothing should happen, and the task should still be in the table.

2.3.1.9 Generate a Gantt Chart

Test Case	Generate a Gantt chart
Test Case Description	To ensure that the correct Gantt chart is generated.
Input	• Log in as a manager • Load a project • If there are no tasks, create and add new tasks by using the task creation screen • Once there are tasks for the project, click on the "Generate Gantt Chart" button
Output	A Gantt chart with the correct task information appears.

2.3.2 Scope 2: Member-Specific Functions

Member-specific functions are those functions which can only be performed while the program is being used by a member. These functions constitute only a small part of all the system's functions and so testing done in this section is accordingly proportionately small.

2.3.1.1 Login as a Member

Test Case	Login as a member
Test Case Description	To ensure that entering Member credentials will enable the user to log in a a member.
Input	• Enter a member's ID and corresponding password • Click on the button "Log in"
Output	The welcome screen is displayed, with the member's name.

2.3.2.2 View Assigned Tasks

Test Case	View assigned tasks
Test Case Description	To ensure that a member can correctly view his or her assigned tasks.
Input	• Log in as a member • Have a manager assign you to some tasks • Click on the button "View Assigned Tasks" on the main screen
Output	A table containing all the tasks to which you are assigned should appear.

2.4 Integration Testing

Integration testing is a type of testing in which individual software modules are combined and tested as a group. Having tested the individual functions through unit testing, we can now combine them together into screens and test their integration screen by screen using the Sandwich method, which is a combination of bottom-up and top-down integration testing. An integration test is run as soon as a component is added to the program.

The following list shows which screens were tested:

- Login Screen
- Main Screen
- Project Selection Screen
- Project Tasks Screen

- Assign Members Screen

2.4.1 Login Screen

Test Case 1	Login screen initialization
Test Case Description	To see if whether the screen initializes correctly when the program has been executed.
Result	Login Screen is initialized correctly.

Test Case 2	Login
Test Case Description	To see whether if a user can login successfully after entering the correct credentials.
Result	The user correctly logs in and the Main Screen is shown.

2.4.2 Main Screen

Test Case 1	Create Project Screen initialization
Test Case Description	To see if whether the Create Project Screen initializes correctly when the ”Create Project” button is clicked.
Result	Create Project Screen is shown.

2.4.3 Project Selection Screen

Test Case 1	Project selection
Test Case Description	Select a project from the drop-down menu to see if the description and the start date display are updated accordingly.
Result	The screen correctly displays the info of the selected project.

Test Case 2	Project selection
Test Case Description	Load the selected project by clicking on the ”Load” button. The Project Tasks screen should initialize correctly along with any tasks assigned to the project.
Result	Passed.

Test Case 3	Modify Project screen initialization
Test Case Description	To see if whether the Modify Project screen for the selected project initializes correctly when the ”Modify” button is been clicked.
Result	The Modify Project screen of the selected project is shown.

Test Case 4	Modify Project screen initialization
Test Case Description	To see if the selected project can be successfully deleted by clicking the ”Delete” button and confirming the action.
Result	The project is deleted and does not show up on the Project Selection screen.

Test Case 5	Back to Main Screen
Test Case Description	Clicking on the ”Back” button should bring the user back to the Main Screen.
Result	The Main Screen is displayed.

2.4.4 Project Tasks Screen

Test Case 1	Add Task
Test Case Description	Add a new task to the project by clicking on the Add Task button and filling out the form with the correct info. Wrong entries should display an error message.
Result	A task is successfully created and appears on the Project Tasks Screen.

Test Case 2	Delete Task
Test Case Description	Selecting a task (if one exists) and clicking on the Delete Task button should remove the task from the project.
Result	Passed. The selected task doesn’t appear on the Project Tasks screen.

Test Case 3	Generate Gantt Chart
Test Case Description	Clicking on the ”Generate Gantt Chart” button should open up a window displaying the Gantt chart of the current project.
Result	Passed.

Test Case 4	Generate PERT Analysis
Test Case Description	Clicking on the ”Generate PERT Analysis” button should generate a PERT Analysis of the current project. The table should display all the tasks and their duration, optimistic, pessimistic and estimated times, and the variance.
Result	Passed.

Test Case 5	Generate Critical Path
Test Case Description	Clicking on the ”Generate Critical Path” button should generate a graph displaying the critical path of the current project.
Result	Passed.

Test Case 6	Show Members screen initialization
Test Case Description	Clicking on the ”Show Members” button should display all the members currently assigned to the project and the task they are assigned to.
Result	Passed.

Test Case 7	Back to Project Selection Screen
Test Case Description	Clicking on the ”Back” button should bring the user back to the Project Selection Screen.
Result	Passed. The Project Selection Screen is now on display.

2.4.5 Assign Members Screen

Test Case 1	Assign a member to a task
Test Case Description	By entering a valid task ID and selecting member from the drop-down list the manager should be able to assign the member to the task.
Result	Passed. The assigned member along with their respective task appear on the Project Members screen.

Test Case 2	Cancel assigning member
Test Case Description	Clicking on the ”Cancel” button should bring the user back to the Members Screen.
Result	Passed. The Member Screen is now on display and no member has been assigned to any task.

2.5 User Interface Testing

User interface testing involves checking if the graphical user interface of the program is correctly handling interactions from the user, as per requirements. The program consists of a series of windows with various user interface components including text boxes, buttons and combo boxes. As such, instead of individually testing each component, which would be tedious and redundant, the testing process will instead consider separate screens as user interface elements to be tested. The validity of their functionalities will be determined based on the specifications of functionalities that were initially elaborated.

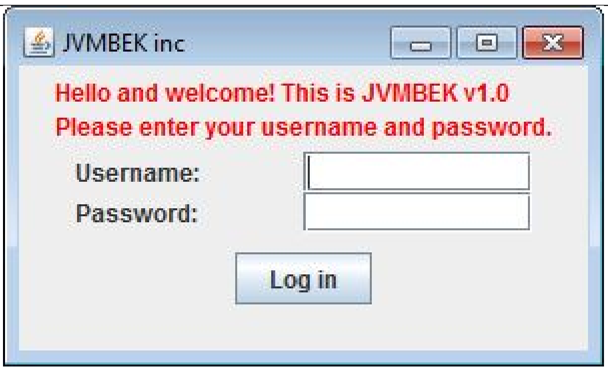
Expected results are taken from the design and specifications document.

The following is a list of user interface elements to be tested:

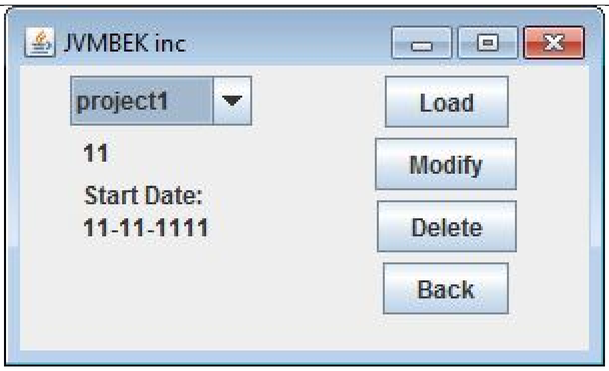
- Login Screen

- Load Screen
- Create Project Screen
- Create Task Screen
- Show Members Screen
- Member Task Information Screen

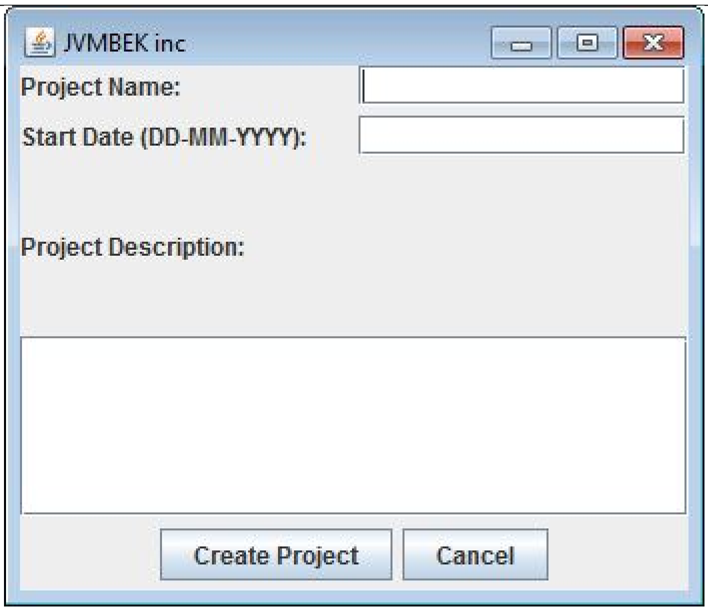
2.5.1 Login Screen

Interface Element	 Hello and welcome! This is JVMBEK v1.0 Please enter your username and password. Username: Password:
Activity Tested	Logging in
Input	A username and a password.
Expected Results	When the user clicks on the "Log in" button, the system should check the validity of the entered input. If an invalid combination of username and password was given, then an error message should appear saying so. Otherwise, the user should be able to log in successfully, and a welcome message should be displayed.
Obtained Results	Upon attempting to log in with invalid input, an error notification appears with the message "The username or password you entered was invalid. Please try again." With valid input, login is successful, and the user is brought to a main screen with a welcome message which includes his or her name.

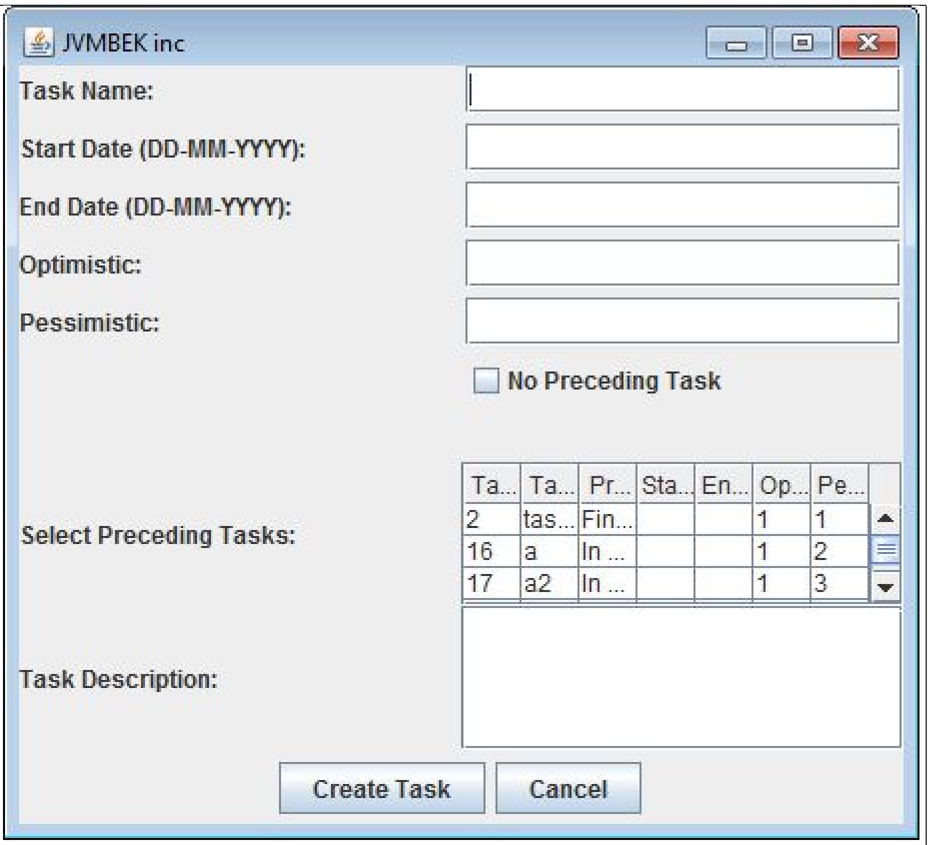
2.5.2 Load Screen

Interface Element	 11 Start Date: 11-11-1111
Activity Tested	Loading a project
Input	A selected project.
Expected Results	After the manager clicks on the button ”Load”, the system should present a list of stored projects from the database. Then, upon selecting a project from the list, the system should load the project into the program.
Obtained Results	Clicking on the ”Load” button will cause the system to find and fetch from the database the project that is currently selected in the combo box. The project is then loaded.

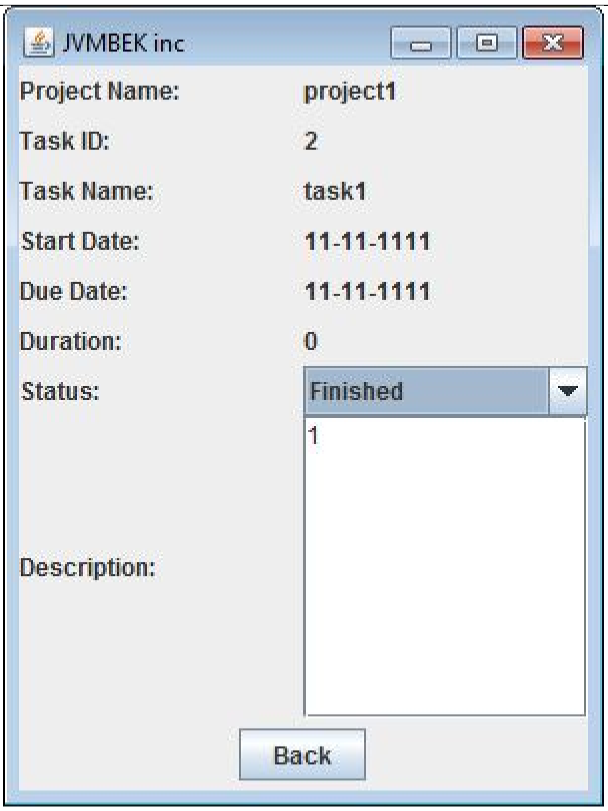
2.5.3 Create Project Screen

Interface Element	 JVMBEK inc Project Name: Start Date (DD-MM-YYYY): Project Description: Create Project Cancel
Activity Tested	Creating a project
Input	Project information.
Expected Results	After entering a project name, start date and end date, and clicking on the button ”Create Project”, the system should create a new project in the database with the given information and load it in the program.
Obtained Results	Upon clicking on the ”Create Project” button after having entered a valid project name and start date, the system will generate a new project with the given input in the database, but will not load it, instead relocating the manager to the main screen. Entering improper input for either the project name or start date fields will cause an error message to be displayed, notifying the manager to rectify the problem.

2.5.4 Create Task Screen

Interface Element	 JVMBEK inc Task Name: Start Date (DD-MM-YYYY): End Date (DD-MM-YYYY): Optimistic: Pessimistic: No Preceding Task Select Preceding Tasks: Task Description: Create Task Cancel Ta... | Ta... | Pr... | Sta... | En... | Op... | Pe... 2 | tas... | Fin... | | | 1 | 1 16 | a | In ... | | | 1 | 2 17 | a2 | In ... | | | 1 | 3
Activity Tested	Creating a task
Input	Task information.
Expected Results	After having entered the required task information in the appropriate fields in a presented form, and then submitting the form, the system should generate the new task in the database with the specified input, associate it with the selected project, and reload the updated project to the program.
Obtained Results	After all the required fields for the task have been filled out with valid input and the "Create Task" button is clicked, the system will generate a new task using this input and store it in the database, associating it with the current project. Then, the manager will be returned to the task selection screen. If there is invalid input entered, such as a start date set after the end date, then an appropriate error message is displayed.

2.5.5 Member Task Information Screen

Interface Element	 JVMBEK inc Project Name: project1 Task ID: 2 Task Name: task1 Start Date: 11-11-1111 Due Date: 11-11-1111 Duration: 0 Status: Finished Description: 1
Activity Tested	Viewing assignments
Input	None.
Expected Results	After clicking on the ”View Assignment” button, the member should be displayed a list of tasks which have been assigned to him or her in grid form.
Obtained Results	Upon clicking the button ”View Assigned Tasks”, the member is presented with a table listing all the tasks which have been assigned to him or her, along with some relevant information about each of them, including the project to which the task belongs, and start and end dates.

3 References

- Daniel Sinnig, ”Introduction to Software Testing”
  http://users.encs.concordia.ca/∼gregb/home/PDF/comp354-testing-intro.pdf
  (Current July 29, 2014)

- Wikipedia, ”Software Testing”
  http://en.wikipedia.org/wiki/Software_testing
  (Current April 1, 2015)