Expensify 360

Software Design Specification Document

Team 5

Abbi Mckann
Dominic Pasquarelli
Christian Sifuentes
Eduardo Trevino Estrada

Version: 1.2 **Date: 12/15/2021**

Table of Contents

1 Introduction

1.1 Goals and objectives

To create a business expense-tracking web application with data visualization.

1.2 Statement of system scope

The software will allow managers to create organizations, projects, and users. Managers will have expense approval and denial authority, be able to delegate project leadership to users and have access to expense summaries and interactive timeseries charts. The software will allow users to generate expense requests, categorize the expense, and include optional supporting documentation (as images) that will be viewable by their manager.

2 Architectural Design

The system follows the Django Architecture MVC pattern and is composed of Models, Templates, and Views.

Models generally map to a dedicated database and define a conceptual database design. Models are used by Views (see definition below) to generate content.

Templates dynamically generate HTML. Templates are responsible for defining the appearance of the user interface and passing event information to Views.

Views define the behavior of the user interface; these are functions which accept a web request and return a web response. A View usually serves as an intermediary between one or more Templates and one or more Models but can also simply be responsible for rendering URLs, checking guard conditions, and defining the user interface on-the-fly by passing arguments to Templates.

Within Models, the key subsystems are the Forecasting Model, Expense Model, Model Managers, and the built-in Django User Authentication framework.

The *Forecasting Model* is responsible for defining charts and summary data derived from expense data. The results are rendered by the Dashboard View (see description below).

The *Expense Model* defines structured expense entry data passed to it by the Expense View. These data are used by Views to render granular expense information and by the Forecasting Model to generate charts and summary data.

Model Managers, serving as the link between program and database management system, query and update data from the database associated with their respective Models.

The *Django User Authentication* framework is responsible for user account logic, manages user groups, user permissions and cookie-based user sessions. Views implicitly use the framework for session information, and explicitly for authentication, determining content rendered to users based on user permissions and groups.

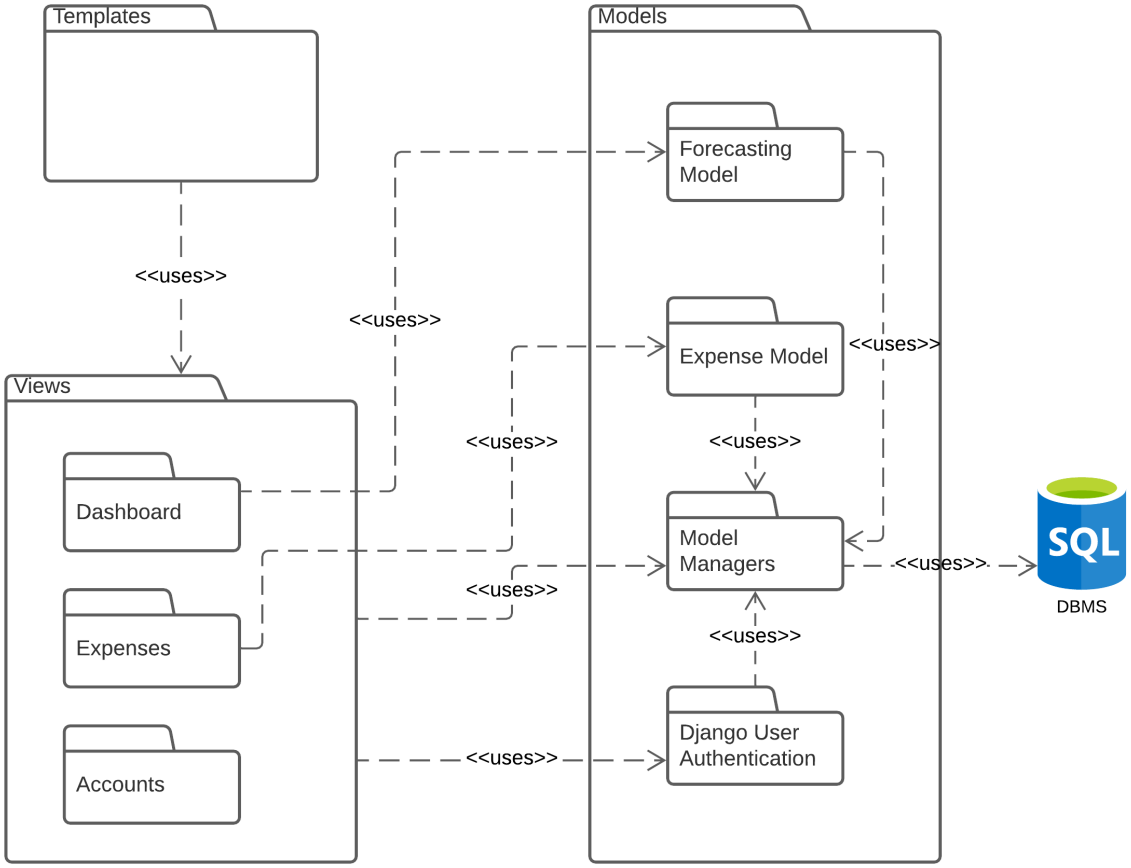
The *Dashboard View* handles the core dynamic content and behavior of the system. The Dashboard uses the Forecasting Model to render charts and summary data, Models Managers to render user and group information as well as to determine functionality exposed to users based on their roles.

The *Expense View* handles expense entry and approval logic. This View is responsible for rendering data entry form, passing validated data to the Expense Model, and then retrieving and rendering entries to another template.

The *Accounts View* is responsible for passing credentials to Django User Authentication, creating new user accounts, and rendering login and registration pages.

2.1 System Architecture

Figure 1 – Package Diagram



2.2 Design Rational

This architecture was selected secondary to framework selection. Django supports a variation on Models-Views-Controllers, though we were free to decide the decomposition at lower levels. The two primary services rendered by the system are data entry and data retrieval, and those responsibilities are sufficiently distinct that they were delegated to two separate subsystems, Expenses and Dashboard, respectively. Being a web application, authentication is also necessary, so a third subsystem, Accounts, was added. These subsystems use Models for data structuring, storage and retrieval, and that module was decomposed into Forecasting and Expense models. Model Managers are built in, as is Authentication. The most important trade-off considered was complexity: the system is considerably more abstracted than necessary, which is not detrimental in

production, but does slow development, at least initially. The up-front time and effort were considered justified as modularity aids in reusability, and subsystems that require considerable time to develop will contain most of the solutions needed for even dissimilar subsystems developed later. Refactoring is also much more reasonable a proposal with well-abstracted code.

3 Key Functionality design

3.1 The Expense Entry

3.1.1 Expense Entry Use Cases

The user will select which type of expense they would like to enter. This request will then be passed to Django for it to load the appropriate template and return it to the user. From there the user will input the expense data and submit it. This data will be then sent to the controller, and it will validate it against the expense model. If valid, the expense model will generate a query for it to be inputted into the database and trigger a return message to be passed back and displayed to the user.

3.1.2 Processing sequence for Expense Entry

The expense entry page will display a page with 3 buttons giving the user the choice between either mileage, expense, or time entry. Once a choice is made the appropriate form will be rendered. All forms require an organization, submission date, expense date, and project to be entered. Specific entry fields unique to each expense also exist with each one also having a total field to display the cost. After the form submit button is pressed it will validate the data for errors, and either provide an error message or a success message and return the user back to the expense selection page.

Figure 2 - Mileage Entry Sequence Diagram

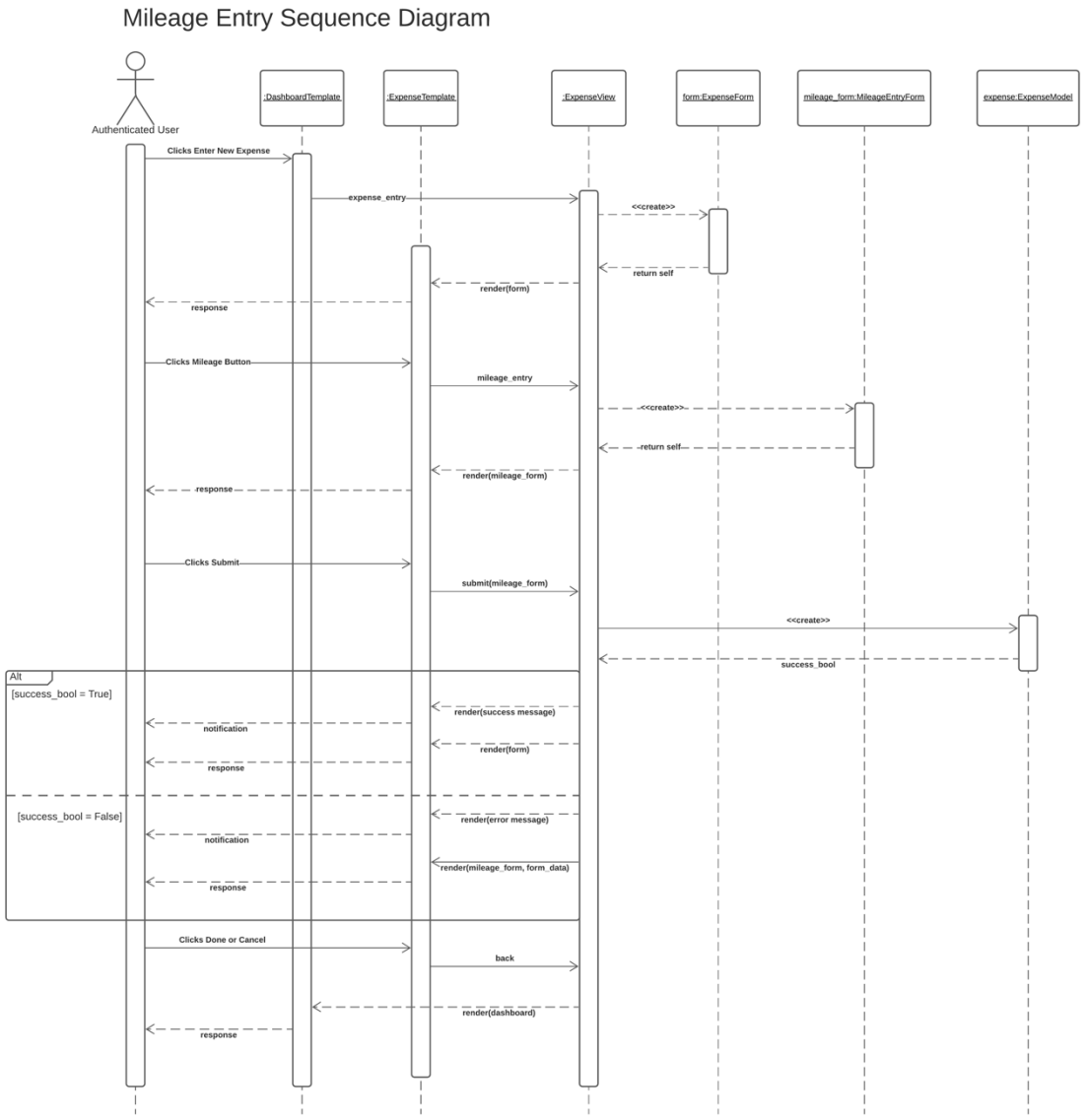


Figure 3 - Expense Entry Sequence Diagram

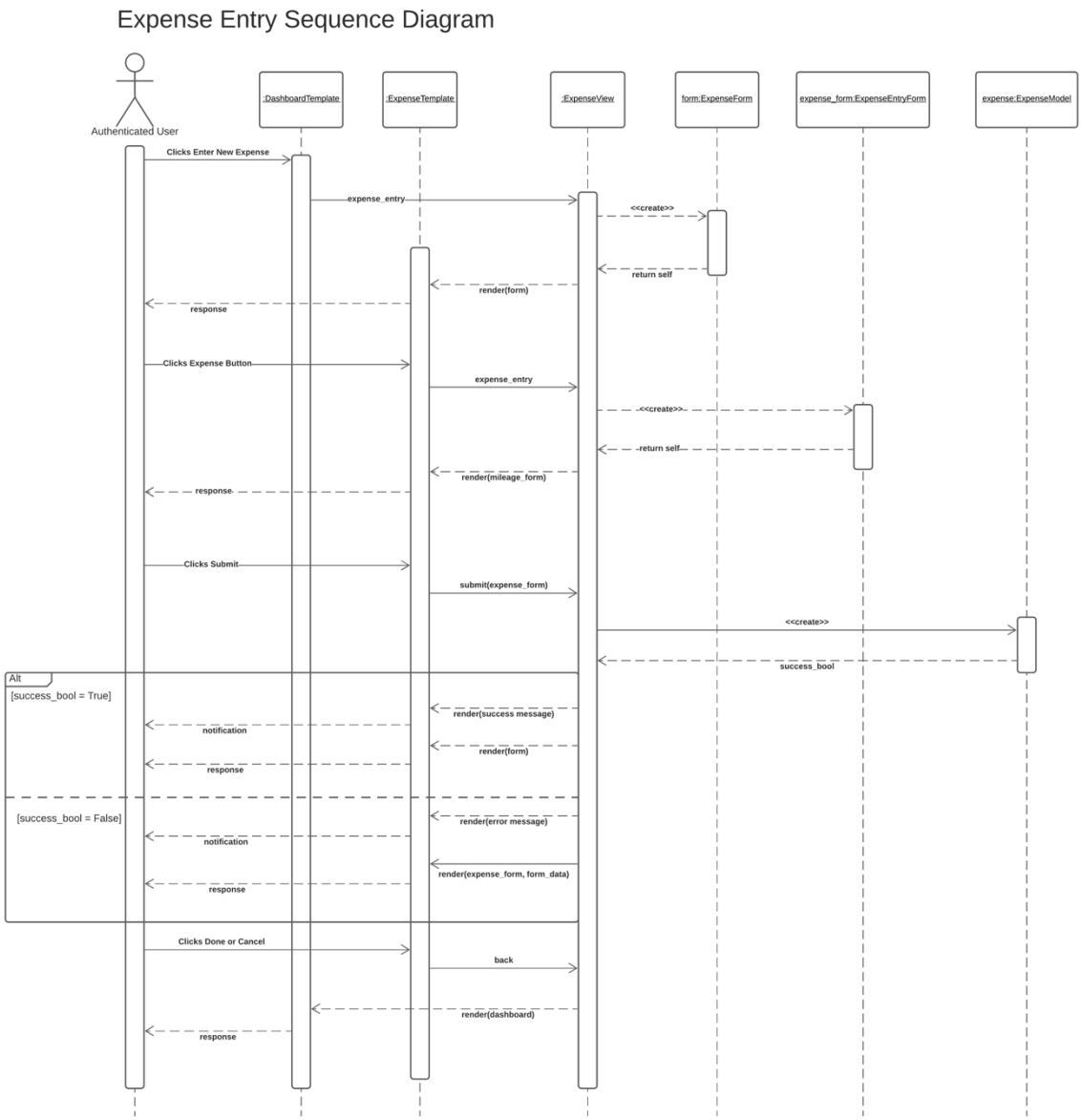
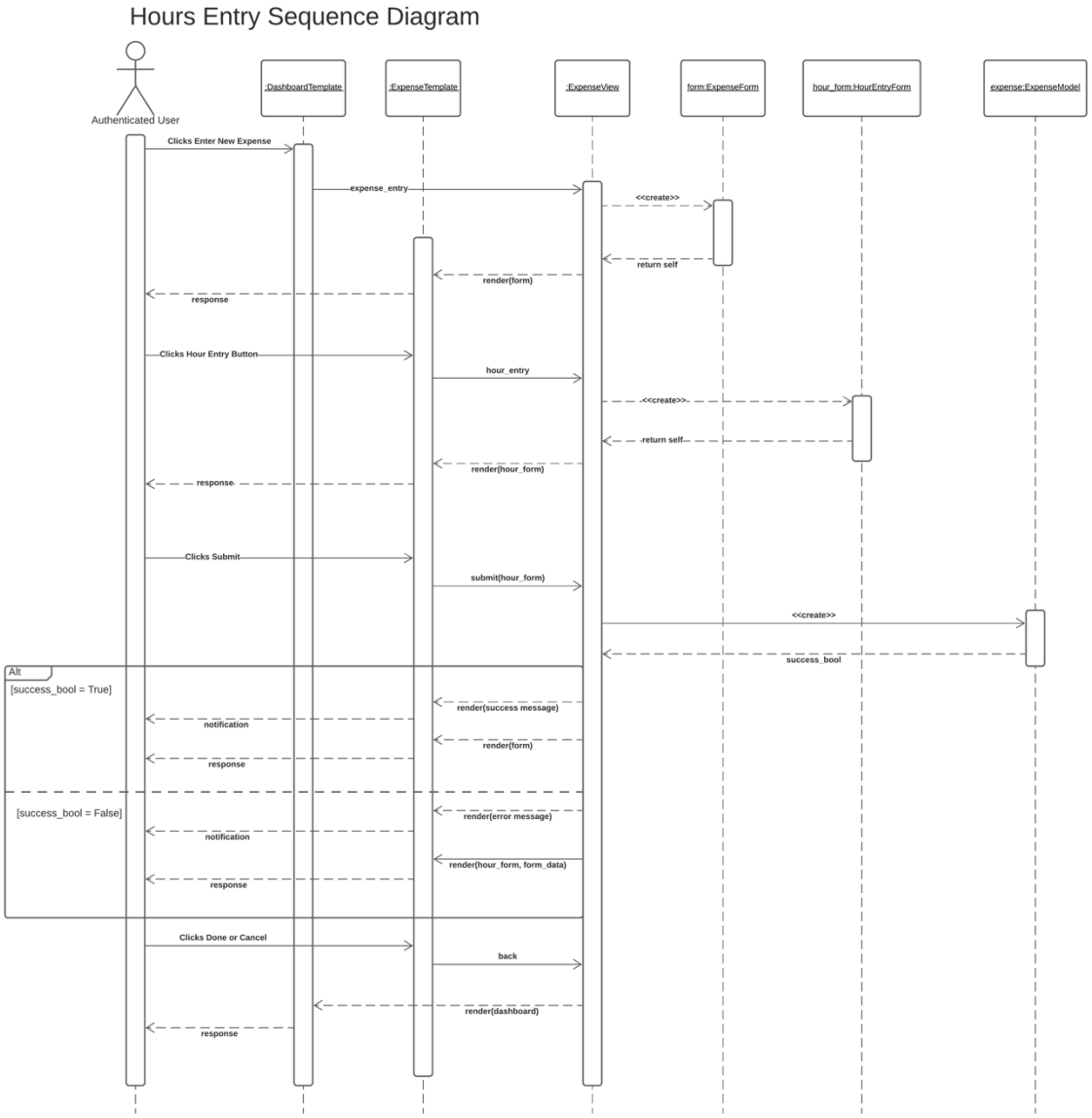
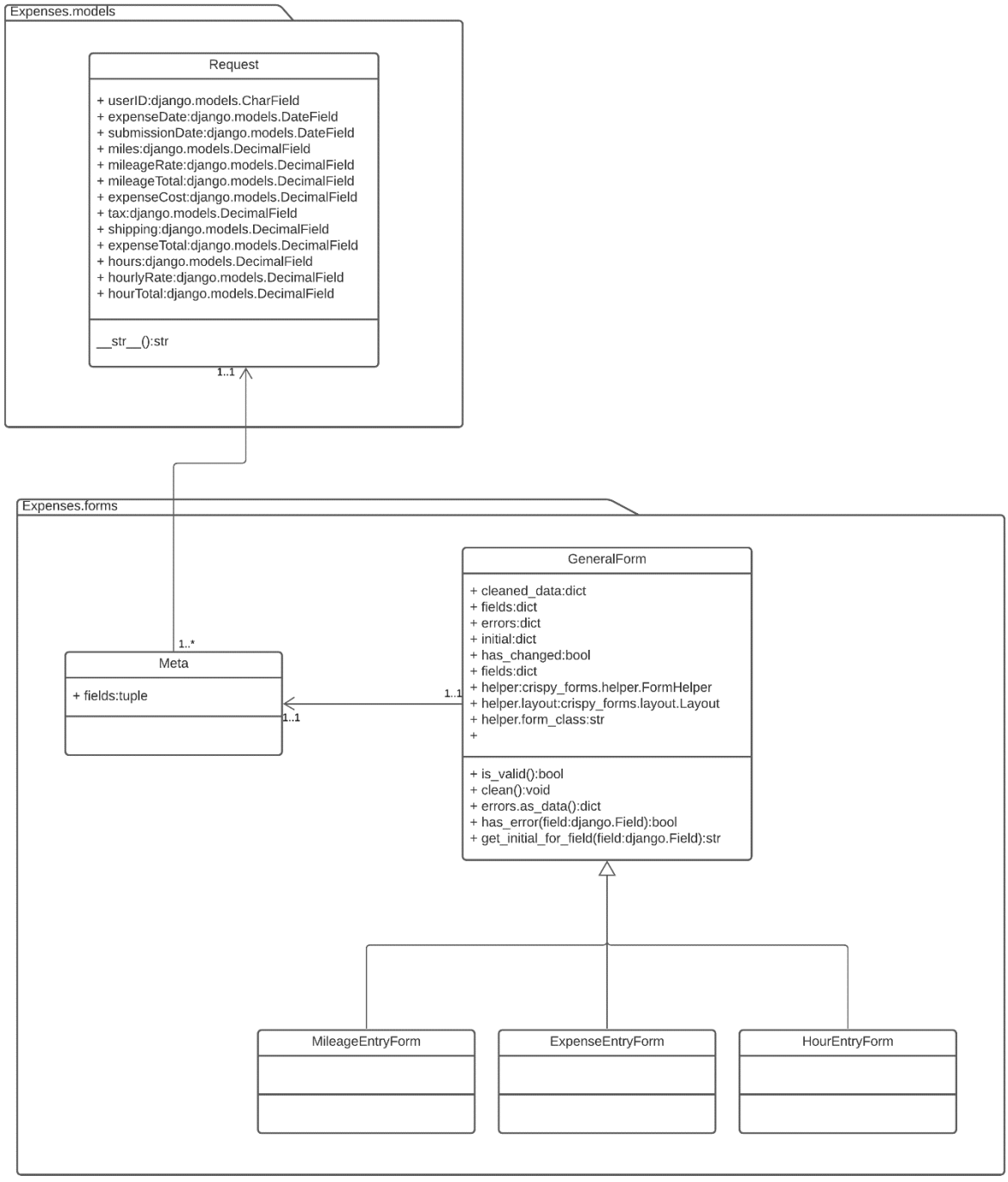


Figure 4 - Hours Entry Sequence Diagram



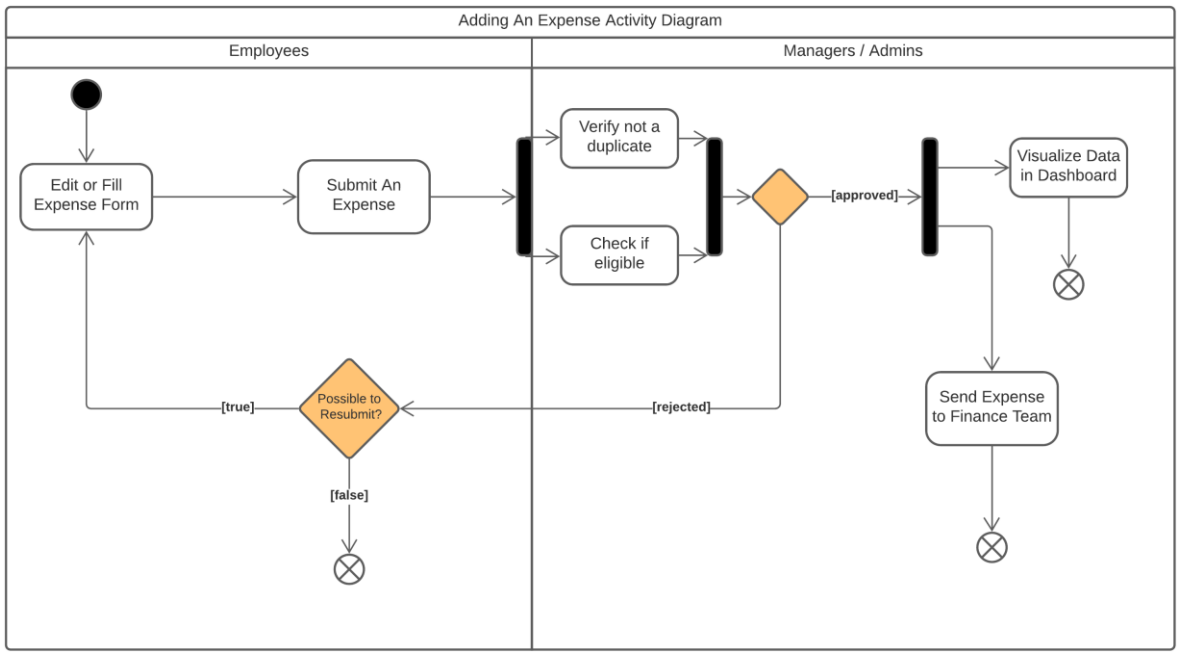
3.1.3 Structural Design for Expense Entry

Figure 5 - Expense Entry Class Diagram



3.1.4 Key Activities

Figure 6 - Expense Activity Diagram



3.1.5 Software Interface to other components

The expense portion of the application interfaces with all other components within the application. This is primarily done through the data logged within the database. The database is used by all aspects of the application and therefore information logged in the expense screen can be accessed and manipulated by the other components. The primary interface within the expense entry is the logging of the user id, project, and organization. When displayed to the user the application gathers the username of the currently logged in user. It then checks what organizations and projects are available to that user and displays only appropriate information.

3.2 The Dashboard

3.2.1 Dashboard Use Cases

The user will come to the dashboard view and if they are a manager-user a query will be made to the database and then converted into various graphical displays. A larger view for each graph can be prompted or the user can move into the expense approval screen to edit or approve employee entered expenses. Within the dashboard a management screen exists which queries the database to display current organizations, projects, and employees. From this menu projects and organizations can be added to the database. Along with employees being assigned to the organizations and projects.

3.2.2 Processing sequence for Dashboard

The dashboard will display several informative graphs on the various expenses in the system. The user can either select one of the graphs to increase its size or use the standard sized graphs. From

the management menu within the dashboard the user will need to select whether they would like to add a project or organization. If creating a project, they need to enter the project name and assign employees to it. Lastly if adding an organization, they will need to add both projects and employees to it.

Figure 7 - Add User Sequence Diagram

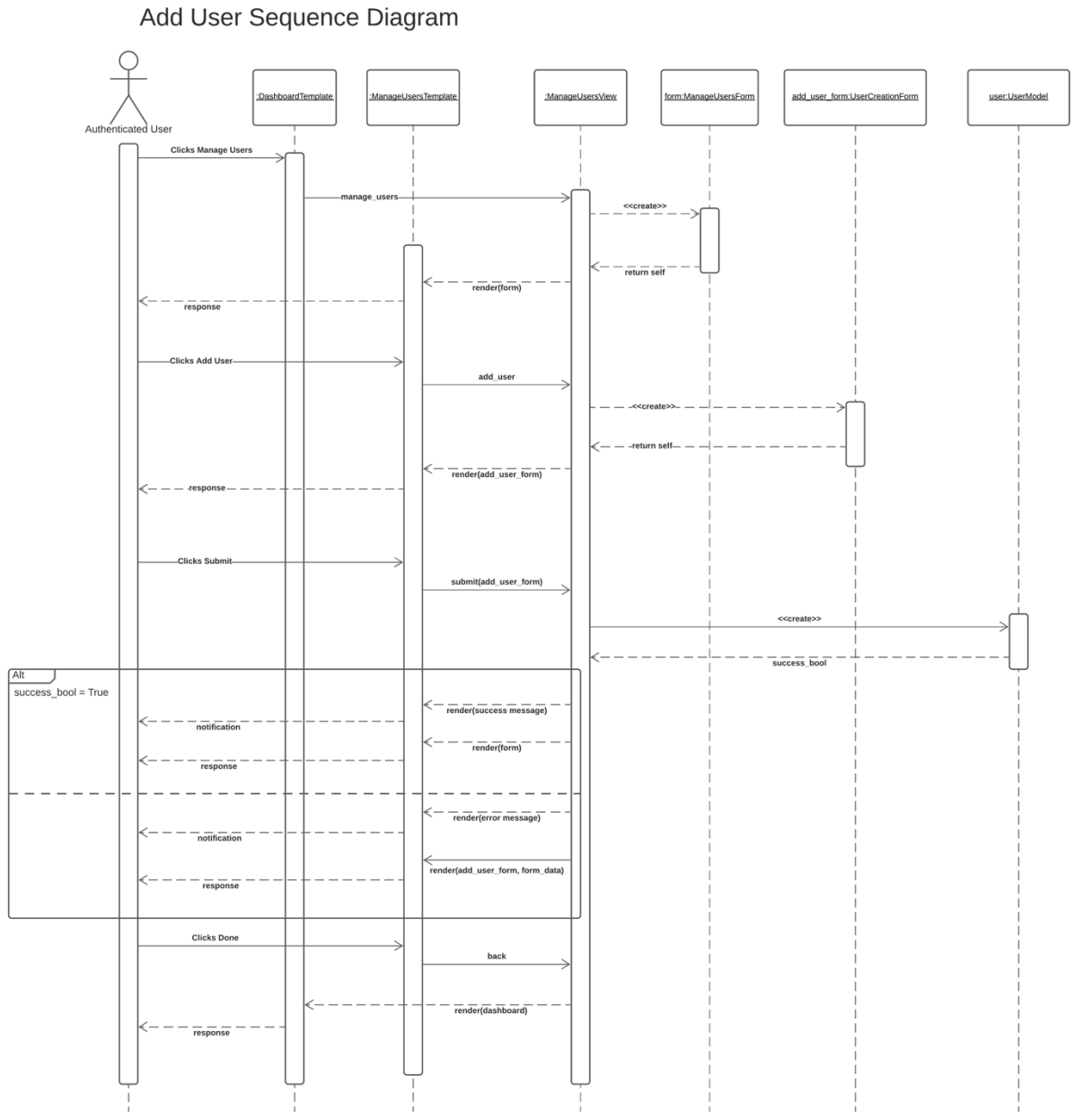


Figure 8 - Remove User Sequence Diagram

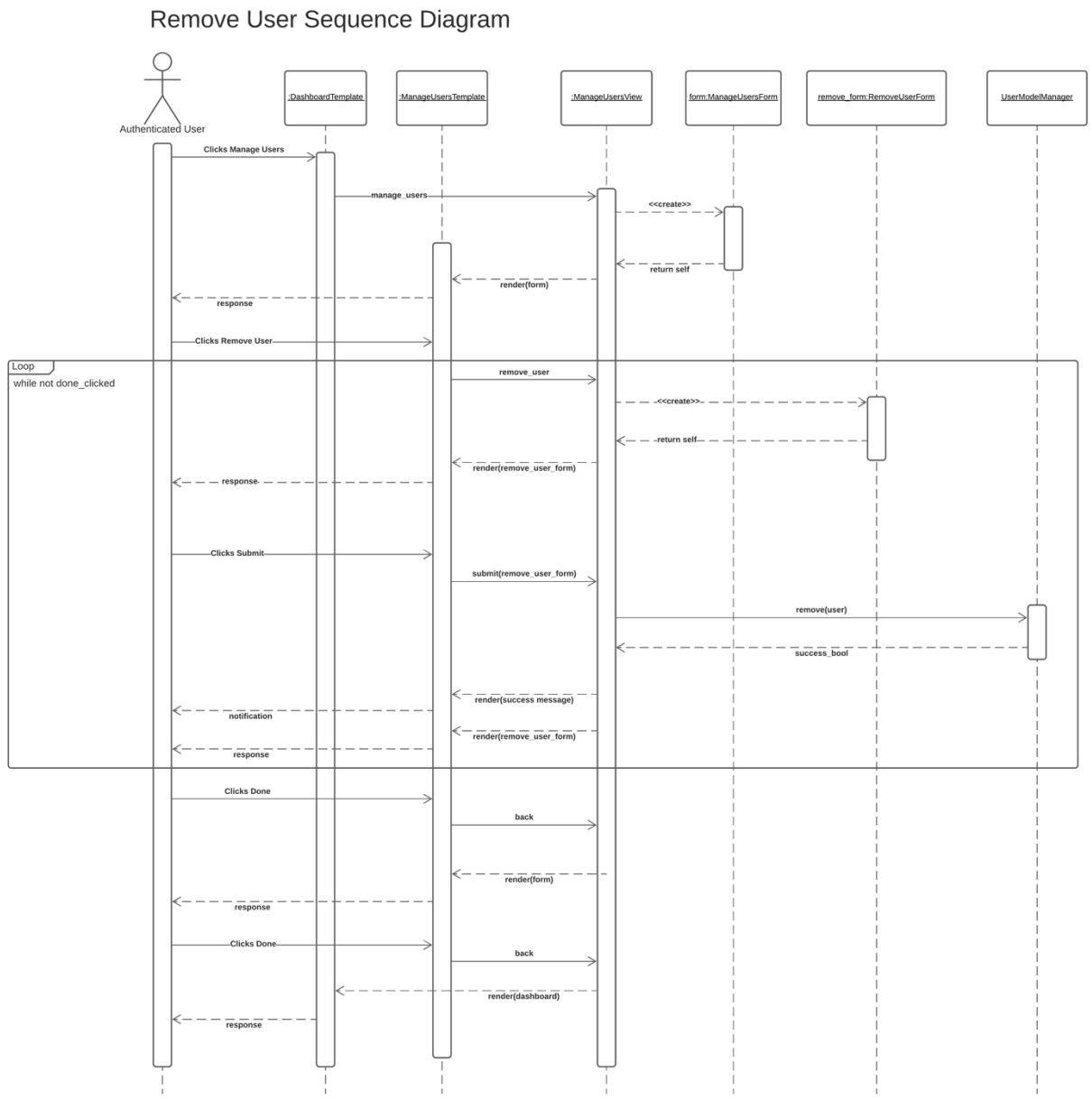
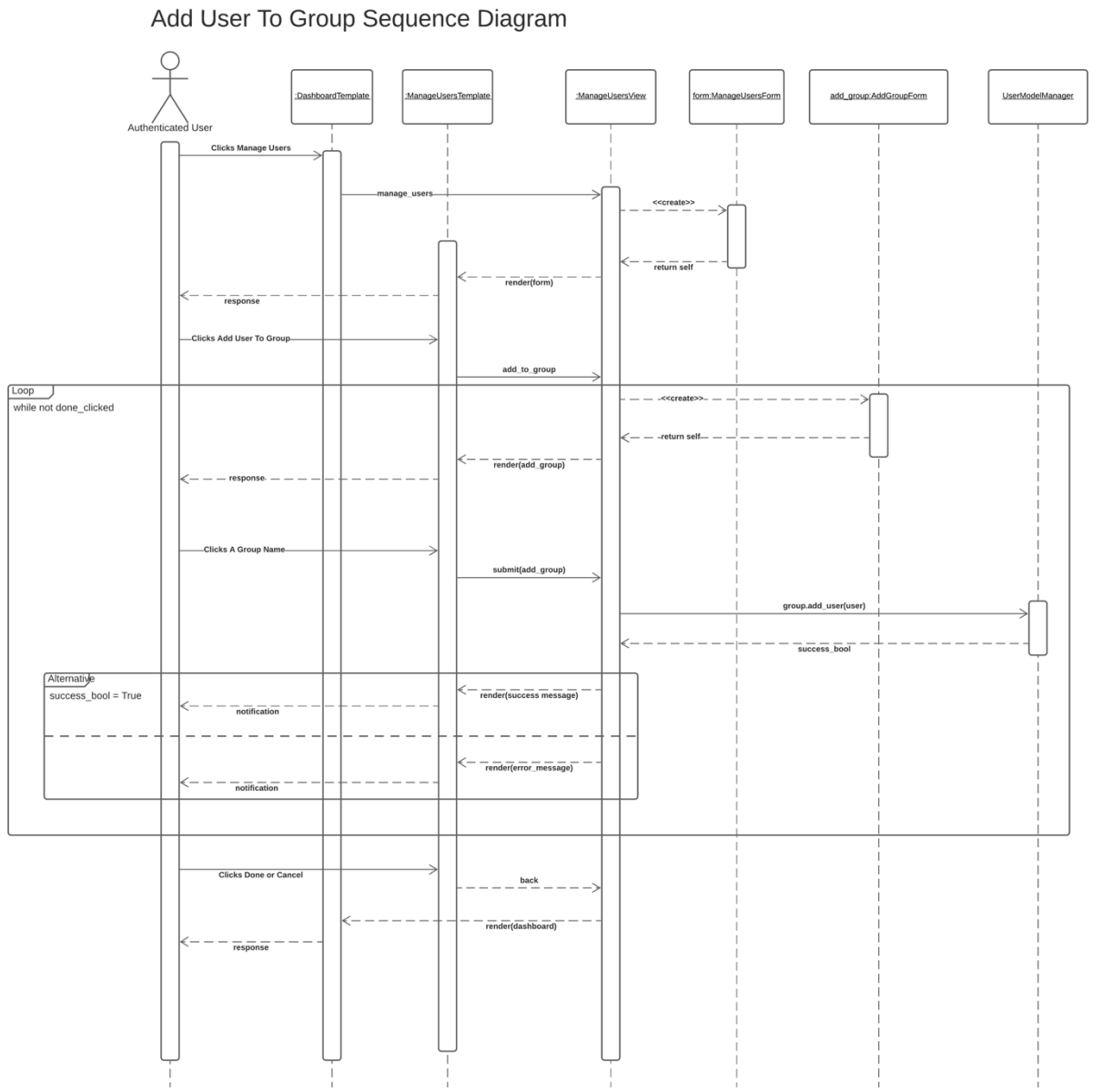
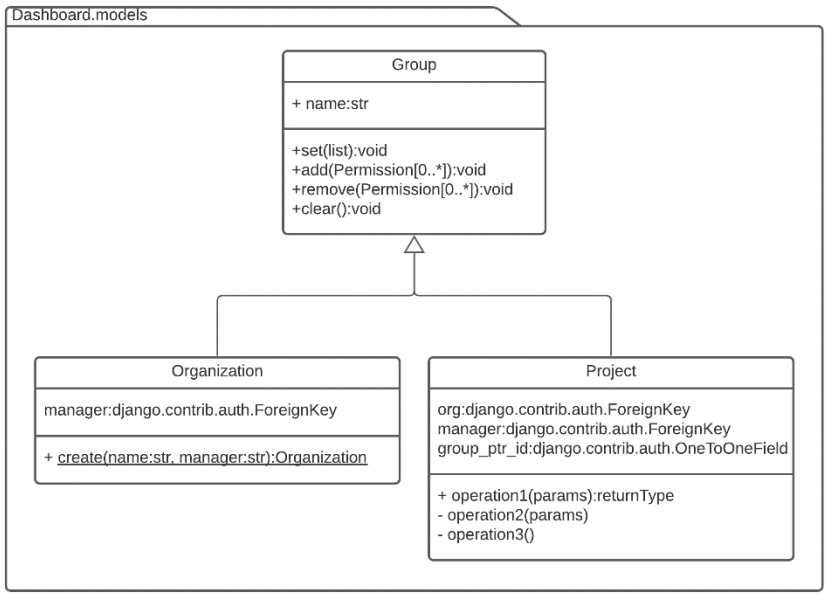


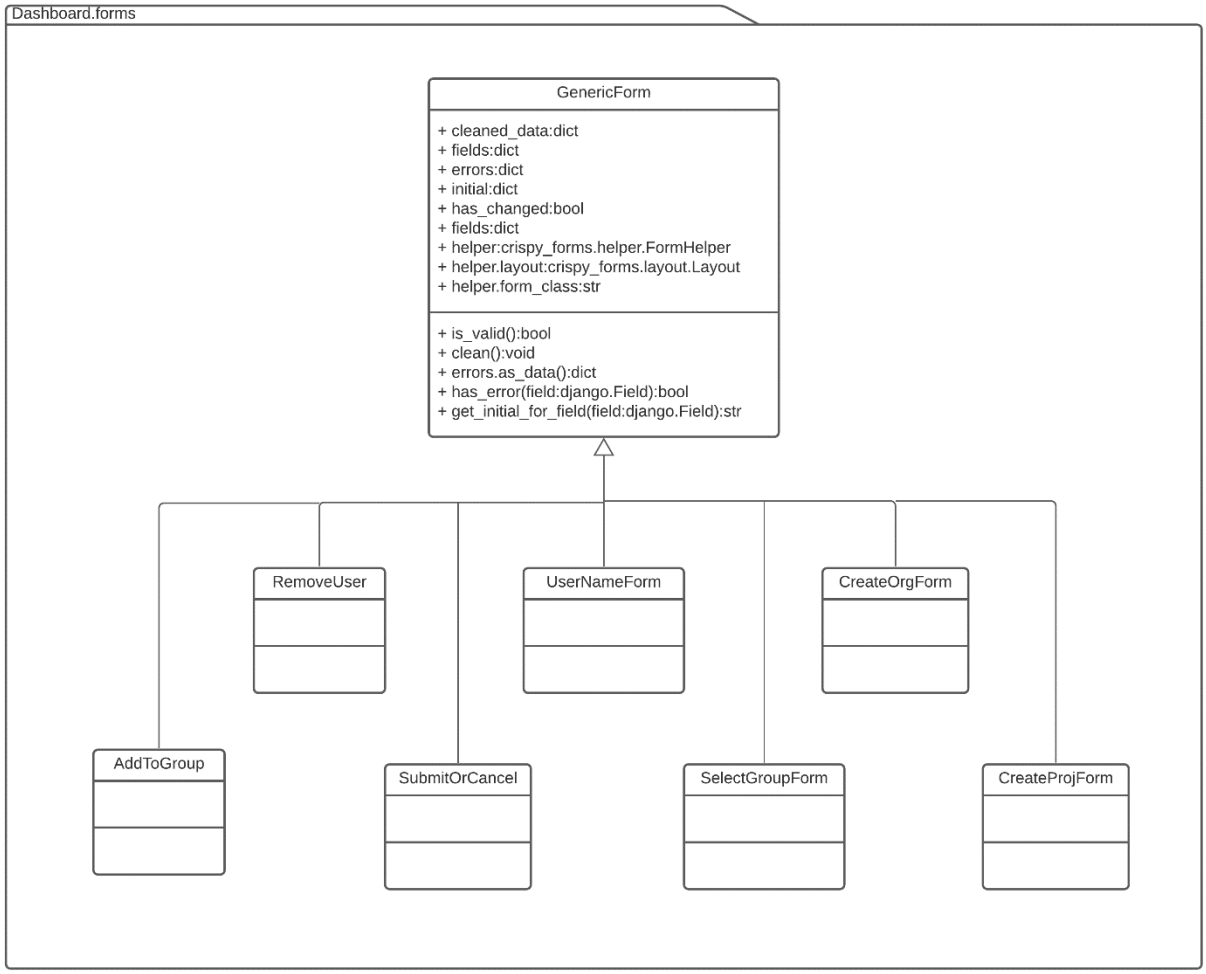
Figure 9 - Add User to Group Sequence Diagram



3.2.3 Structural Design for Dashboard

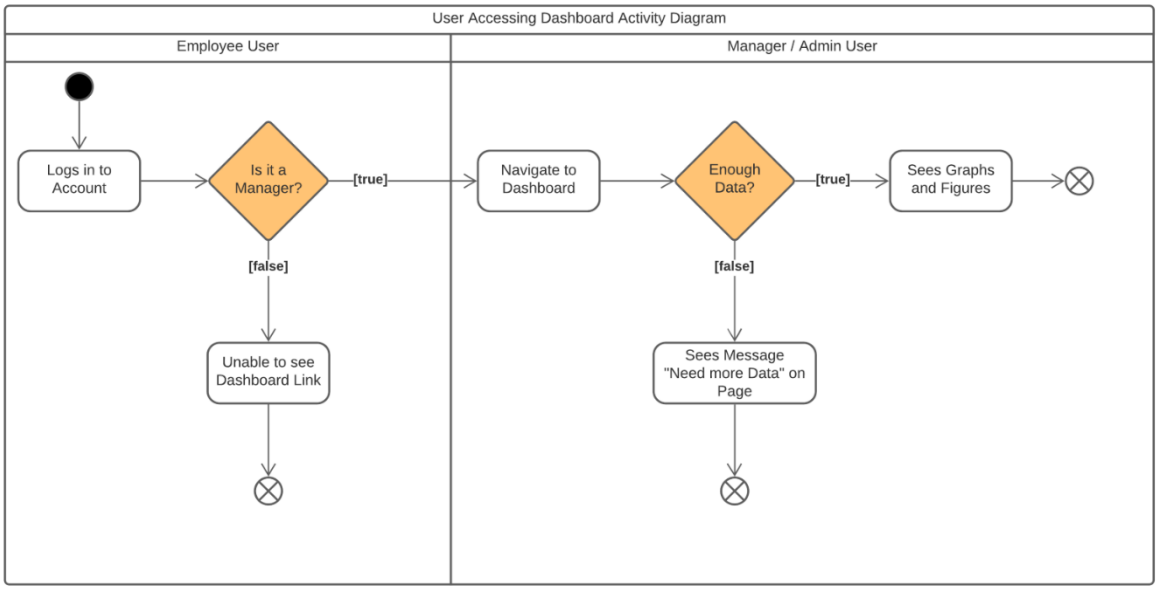
Figure 10 - Dashboard Class Diagram





3.2.4 Key Activities

Figure 11 - Dashboard Activity Diagram



3.2.5 Software Interface to other components

The dashboard page primarily interfaces with other components by its ability to trigger a review of expenses. If a manager user chooses to review or edit an expense it loads it back into the expense view giving the manager, the ability to edit the data originally entered. This then updates the database and in turn the update is reflected in the dashboard.

3.3 The User Account

3.3.1 User Account Use Cases

Several different screens exist for the user account. The first screen is the login. Upon logging in the system will query the database to verify that all account information is correct and if so, load the user onto the main page of the application. If the login information is not correct an error message will be displayed. User accounts can be created in several ways. If an account is created at the log in page it will default to the lowest level user and not have access to any organizations or projects until a manager assigns them. If a user is created from the management panel they can immediately be assigned to an organization and projects.

3.3.2 Processing sequence for User Account

The login page will require both a username and password. Once the login button is entered it will validate the information and display the main login page. If the register account page is requested by the user it will require the user to enter an email address, username, and password. If creating a user from the user management page the same information as the traditional registration page is required along with project and organization information for the user.

Figure 13a - Login Sequence Diagram

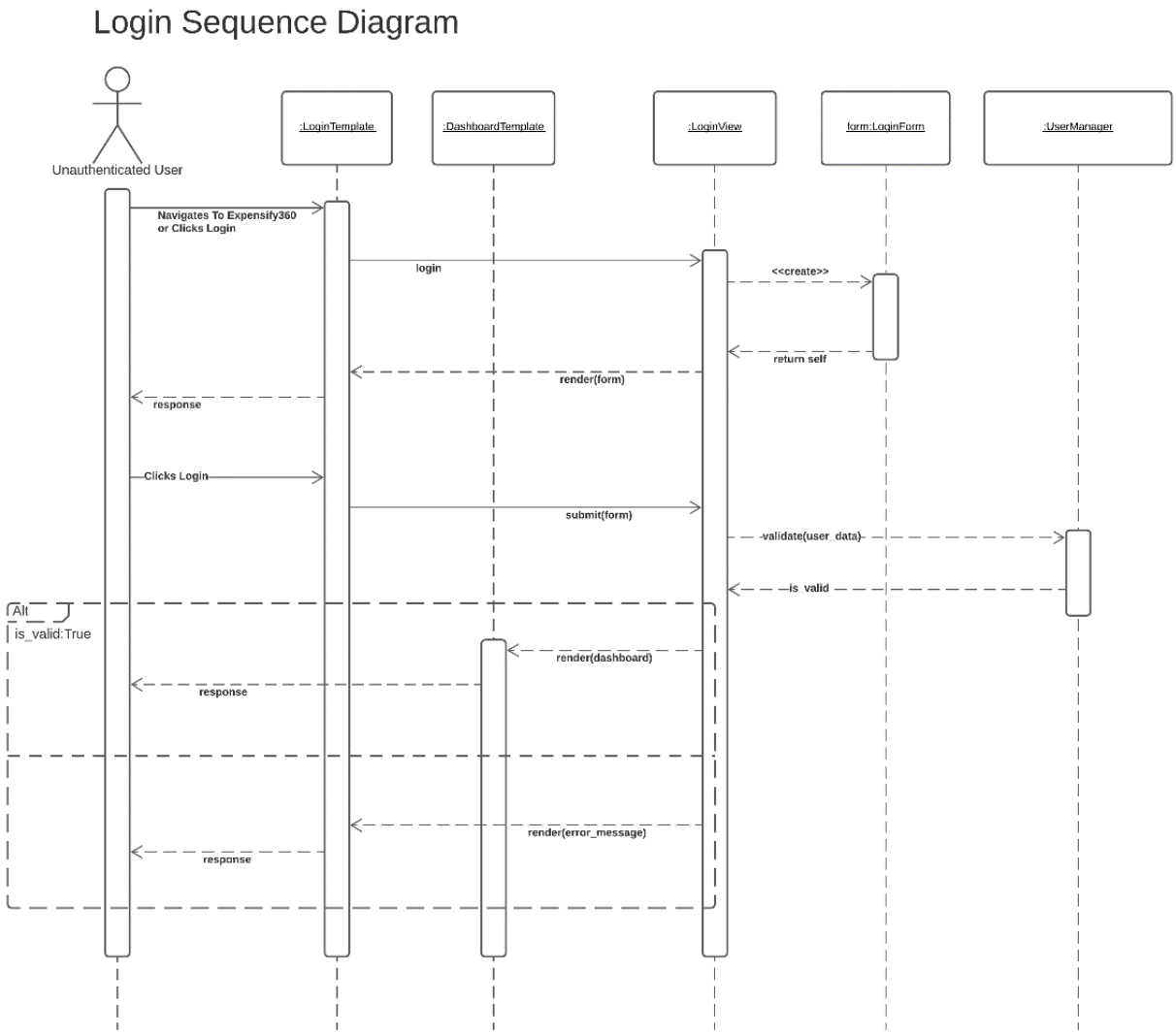
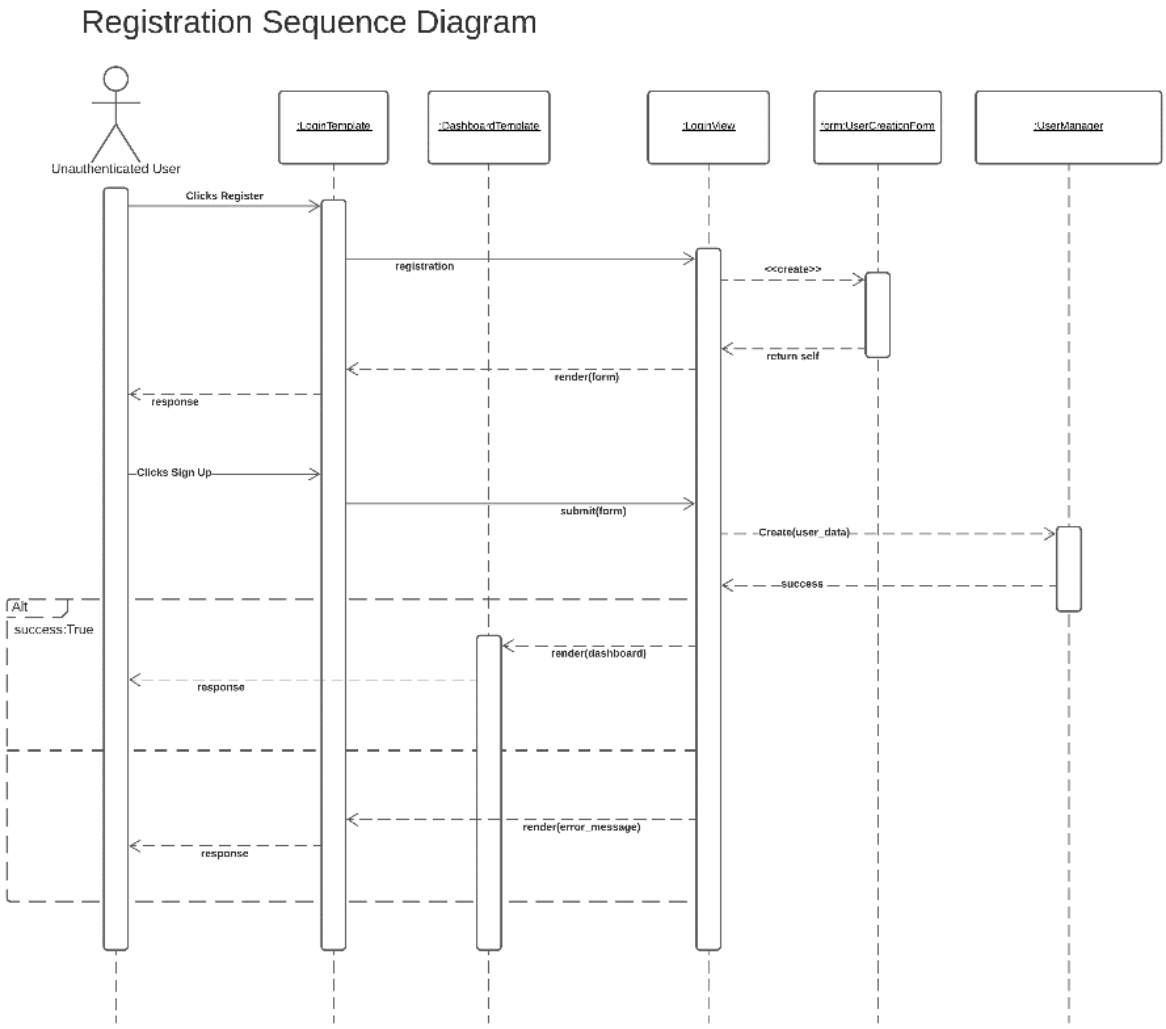
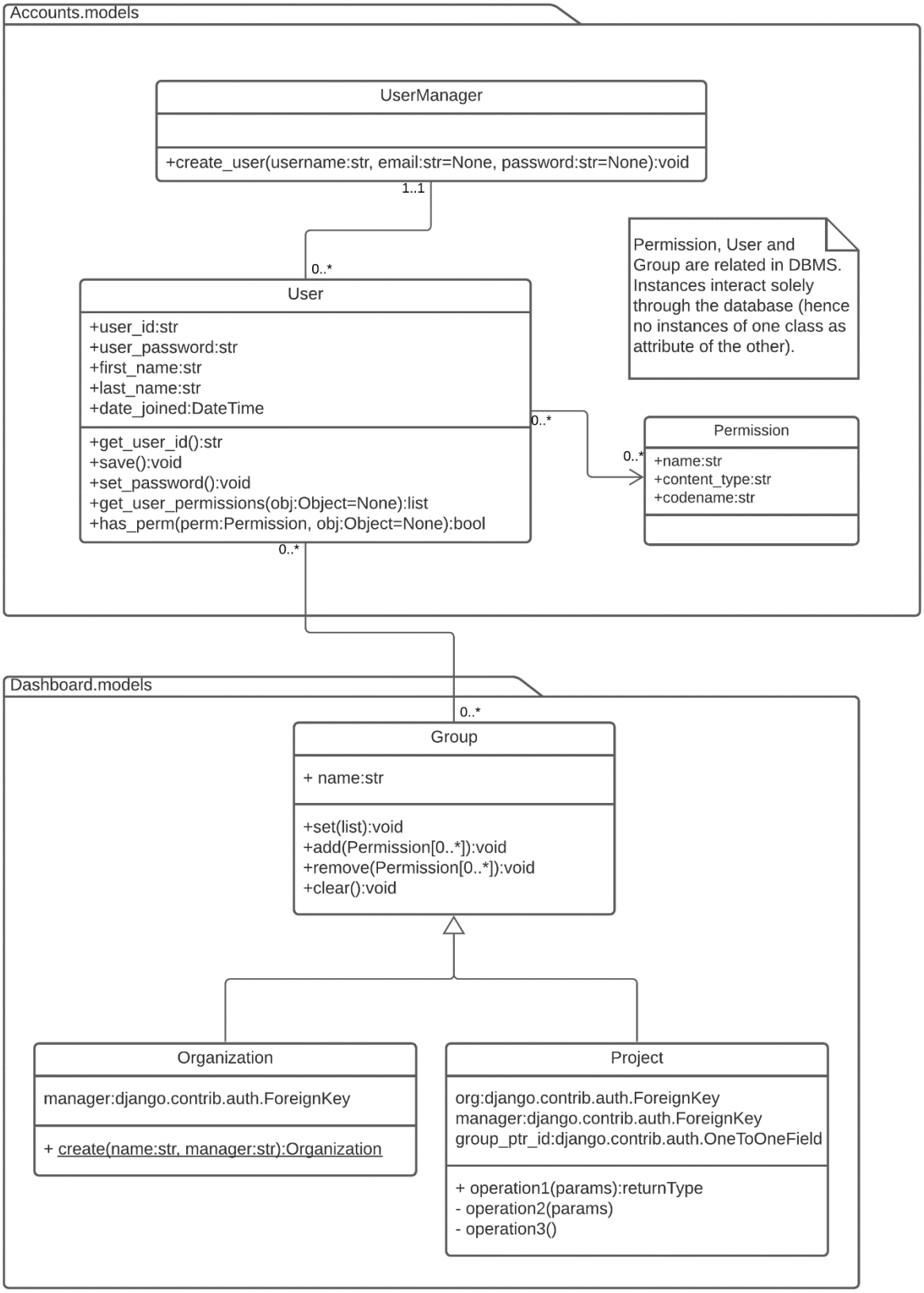


Figure 13b - Registration Sequence Diagram



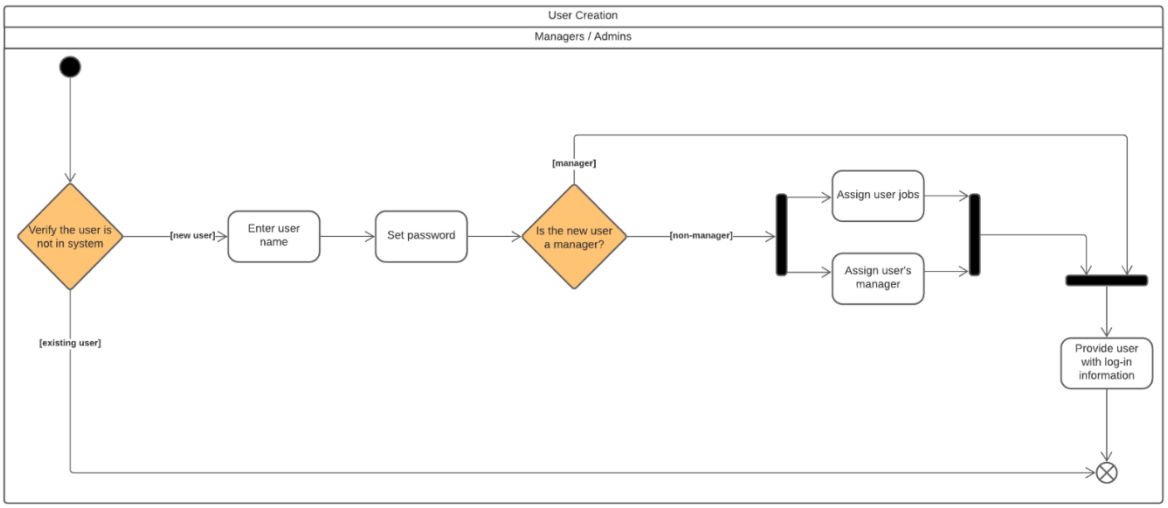
3.3.3 Structural Design for User Account

Figure 14 - User Account Class Diagram



3.3.4 Key Activities

Figure 15 - User Creation Activity Diagram



3.3.5 Software Interface to other components

The user interfaces with almost all pages of the application. The primary way it interfaces is the fact that most pages require the user to be logged in to access them. Areas of the dashboard are locked behind different user levels such as manager or just employee. The expense approval screen is also locked behind a manager-level user.

4 User interface design

4.1 Interface design rules

The user interface follows the standard theme of the open-source CSS framework Bootstrap, version 4.

4.2 Description of the user interface

The user interface (UI) is web-based, allowing users to enter and retrieve data and view transformed data based on their roles.

4.2.1 The Login Page

The login page allows users to enter credentials for access to the app, with a link to registration.

4.2.2.1 Screen Images

Shown here are the login and registration interface screens for the web application.

Figure 16 - Login Page

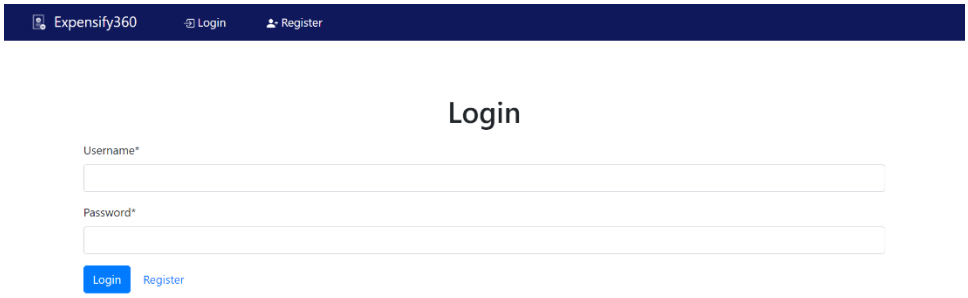
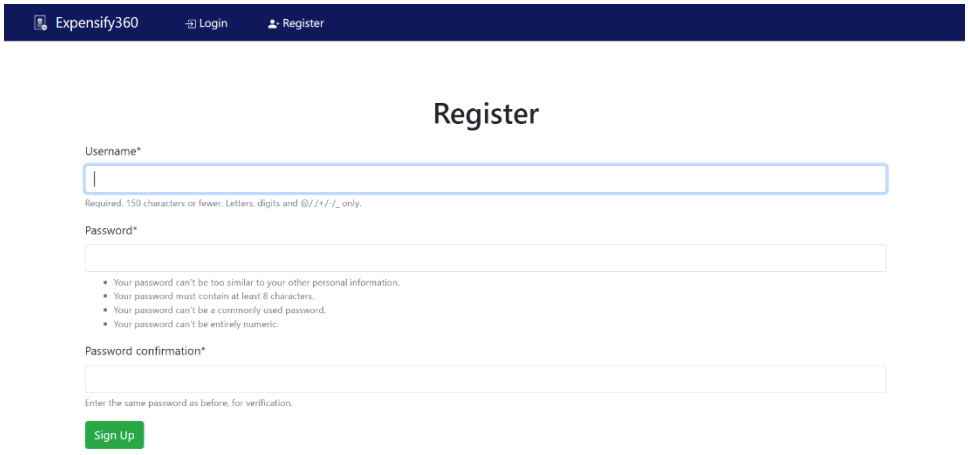


Figure 17 – Registration Page



4.2.1.2 Objects and Actions

The authentication pages will display a modified navigation bar. The dashboard button will redirect to login and there will be links to login and registration pages. The login page will render two fields: username and password. There will be a login button, and unsuccessful login will prompt the user to enter valid information. Successful authentication will result in the user being directed to the dashboard. A register link will also appear on the login page. The registration page will have fields for username, password and password confirmation. Invalid entries will result in the user being prompted to correct the error. Successful registration will direct new users to the dashboard page.

4.2.2 Dashboard Page

The dashboard page is the primary to the user. It will contain links to all key pages and an at-a-glance overview of the user's information.

4.2.2.1 Screen Images

Shown here are the organization and project interface screens for the web application.

Figure 18 – Dashboard Page

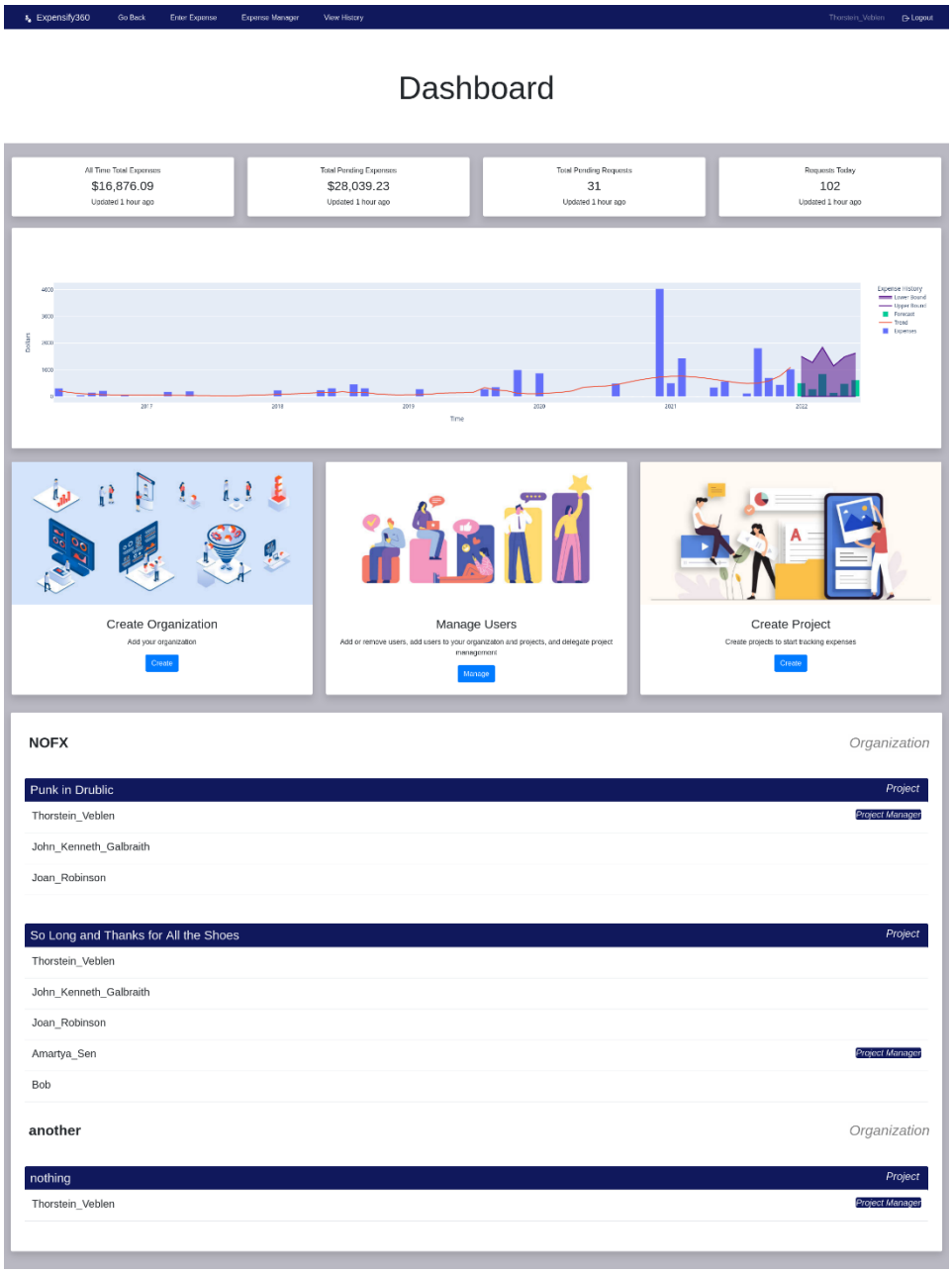


Figure 19 - Organization Creation Page

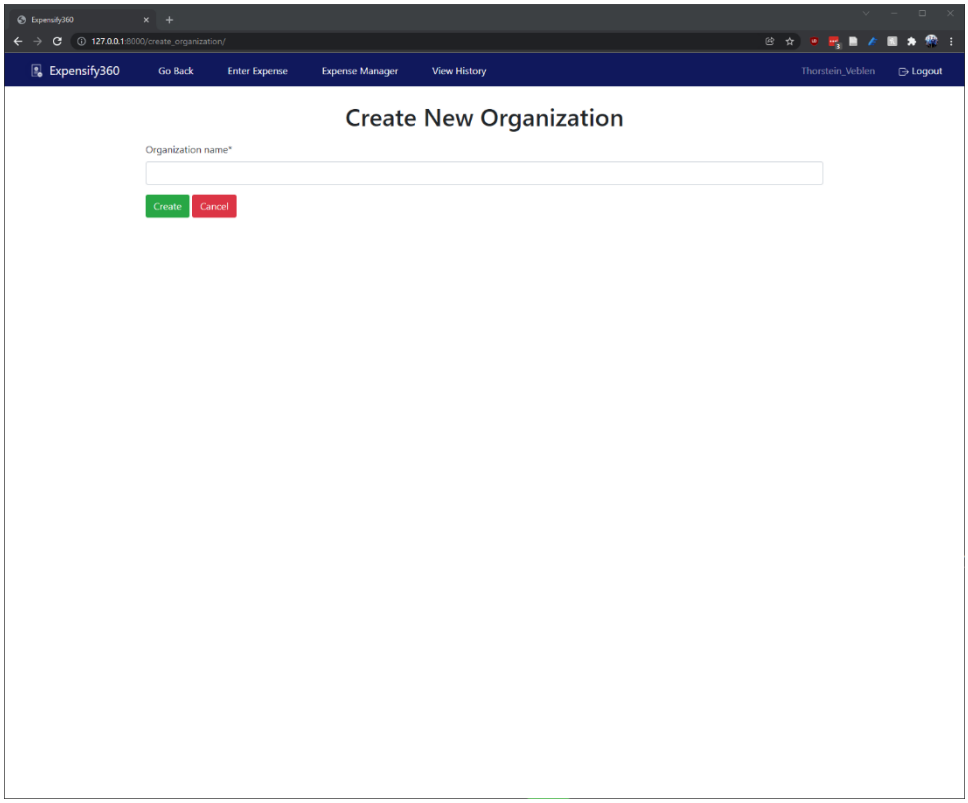
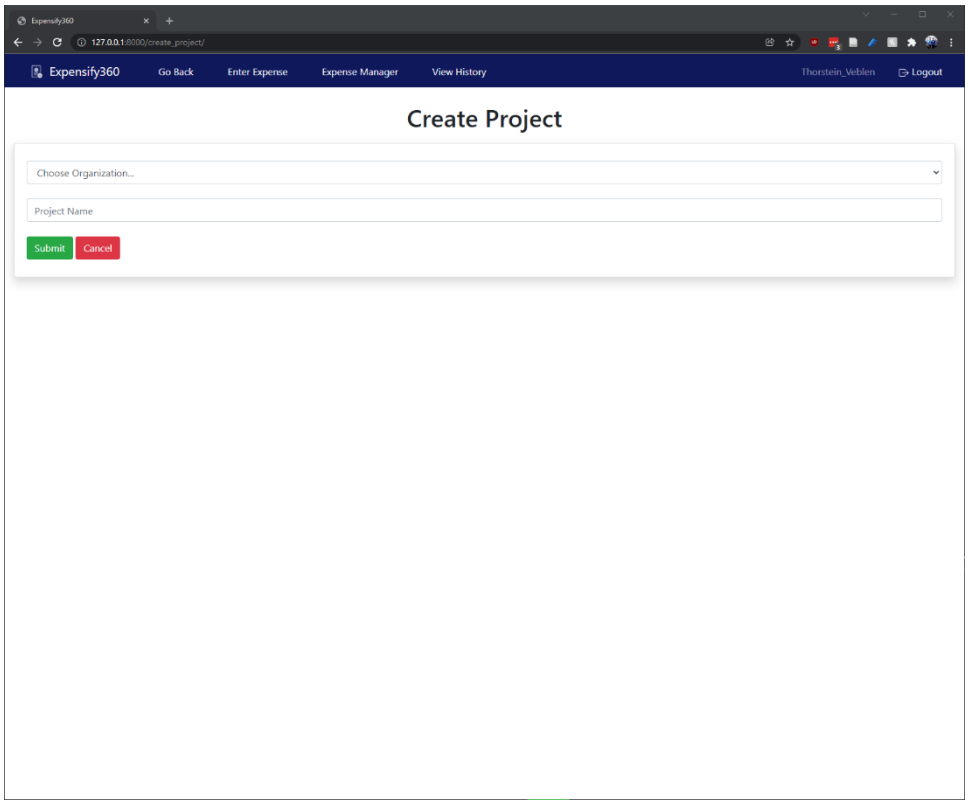


Figure 20 - Project Creation Page



4.2.2.2 Objects and Actions

The dashboard will show different content depending on the user's role. All users will see a navigation bar on all pages, with links to the dashboard, a back button, a logout button, and their username. A manager (organization owner) will see data summaries and visualization, the structure of their organization as well as links to create organizations, manage users, and create projects within existing organizations. A project manager will see project-level data summaries and visualization, the structure of their organization, and a link to manage users assigned to their project(s). An employee-level user will see the structure of their organization.

4.2.3 Expense Entry Page

The expense entry page will give users access to forms for entering expense requests.

4.2.3.1 Screen Images

Shown here are the expense entry screens for the web application.

Figure 21 - Expense Entry Selection Page

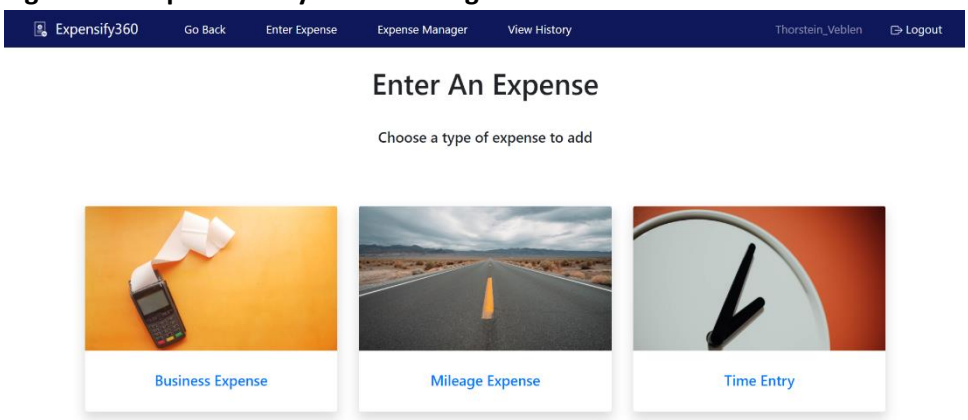


Figure 22 - Mileage Entry Form

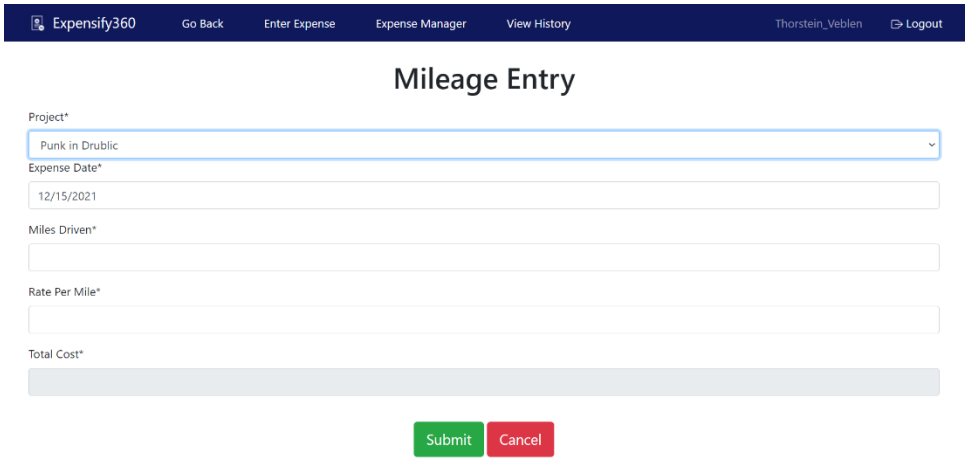


Figure 23 – Business Expense Entry Form

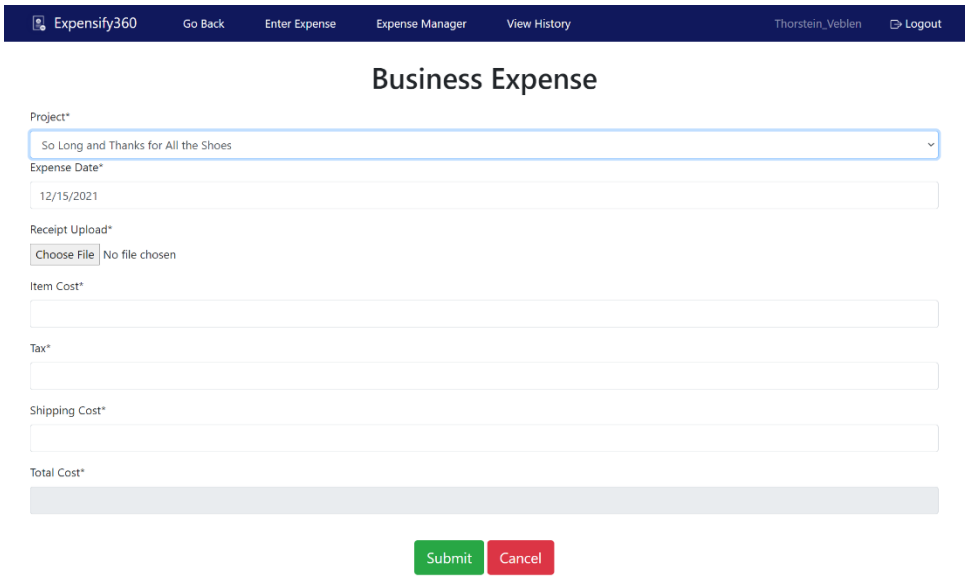


Figure 24 - Time Entry Form (Mobile View)

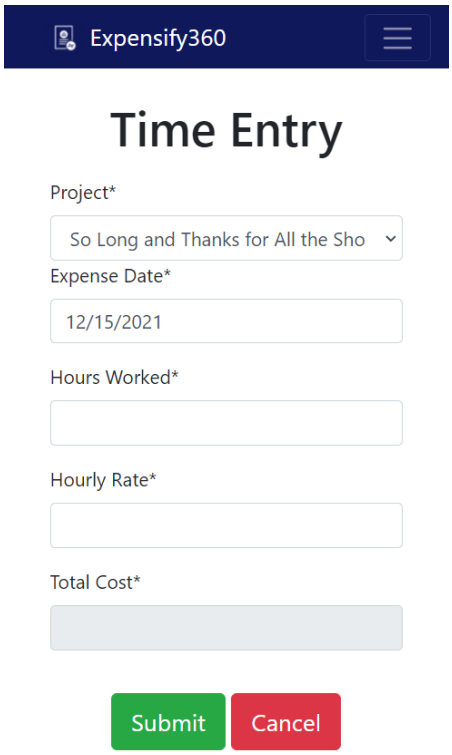
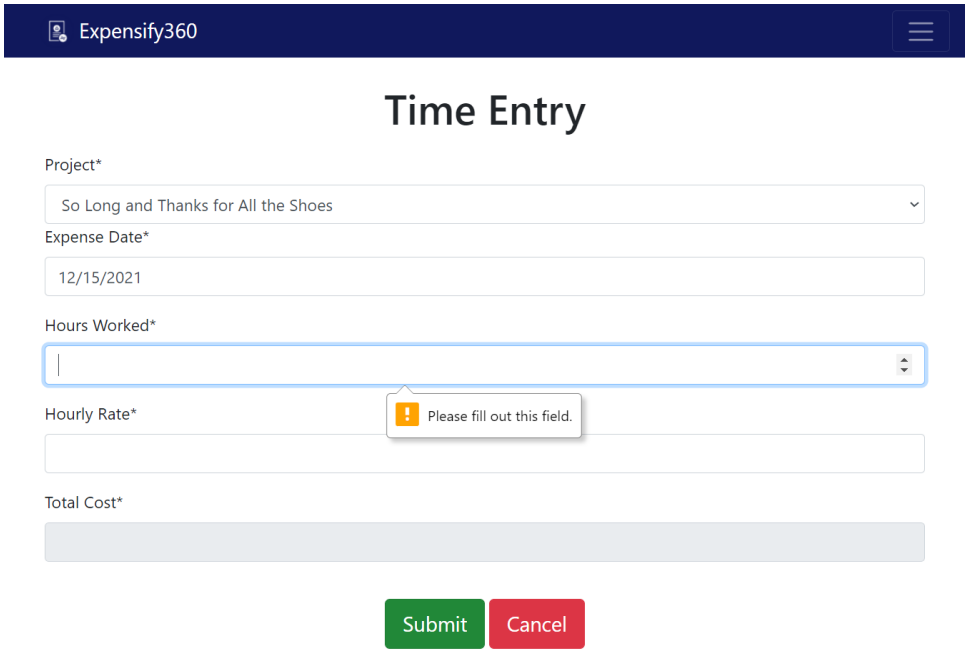


Figure 25 - Entry Form Invalid Field Sample



4.2.3.2 Objects and Actions

On the main expense entry page, the objects are the navigation bar and the three button options. The navigation bar is detailed in 4.2.2.2, and the buttons link to entry forms. On all three forms, the fields are the primary objects. The User ID and Expense Date Fields are automatically filled but not locked. The remaining fields are required, and on submission an empty field will display help text prompting the user to fill out the field. On a successful submission a success message will be displayed in a banner and the user will be returned to the main expense entry page.

4.2.4 User Management Page

The user management page will give the authorized user access to user permissions and groups as well as exposing user creation and deletion functionality.

4.2.4.1 Screen Images

Shown here are the management interface screens for the web application.

Figure 26 - User Management Screen

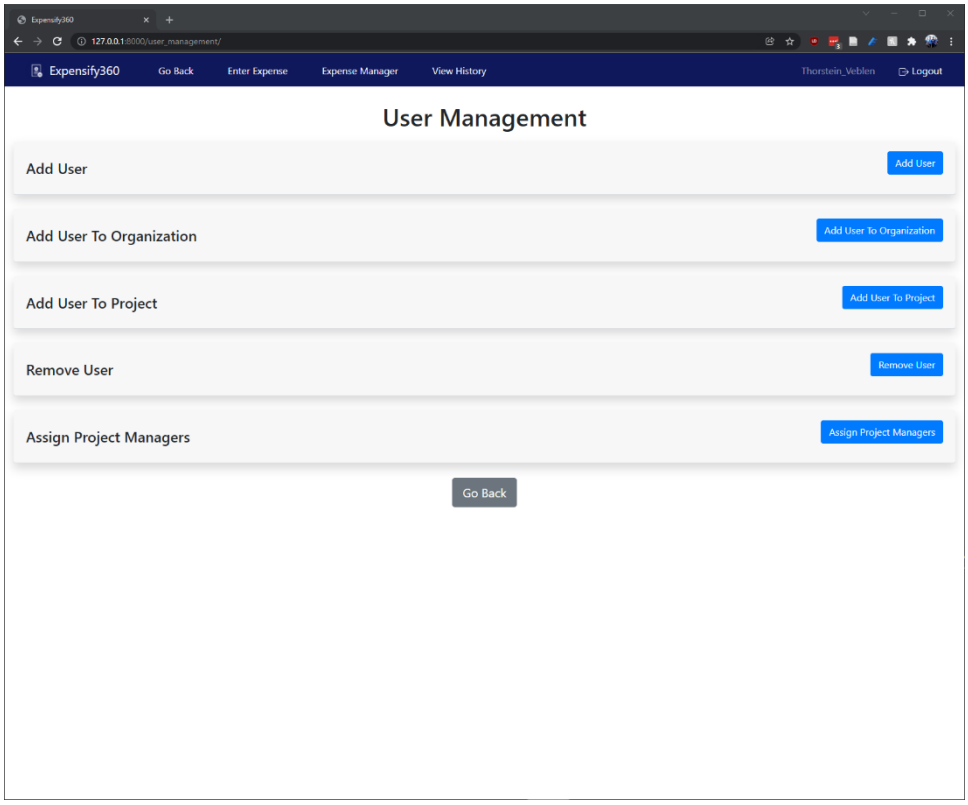


Figure 27 - New User Creation Screen

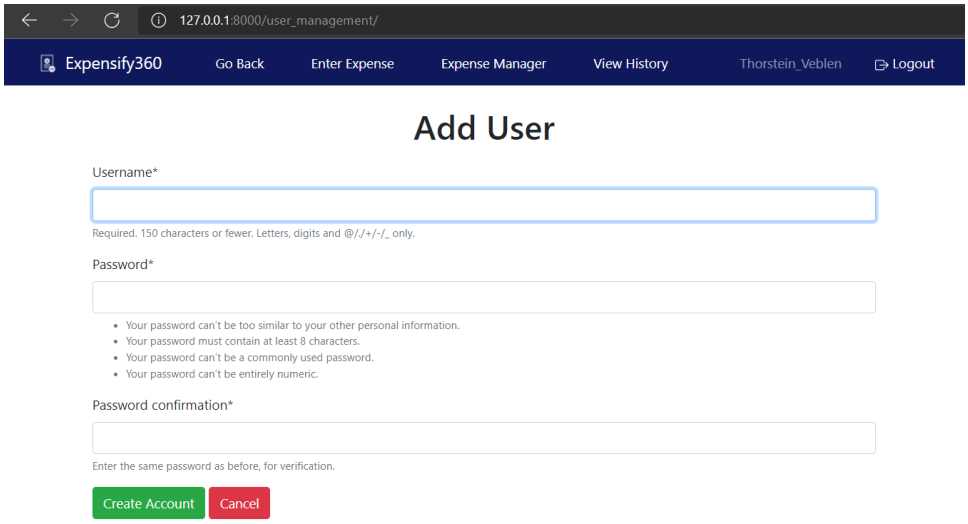


Figure 28 - Organization User Management Screen

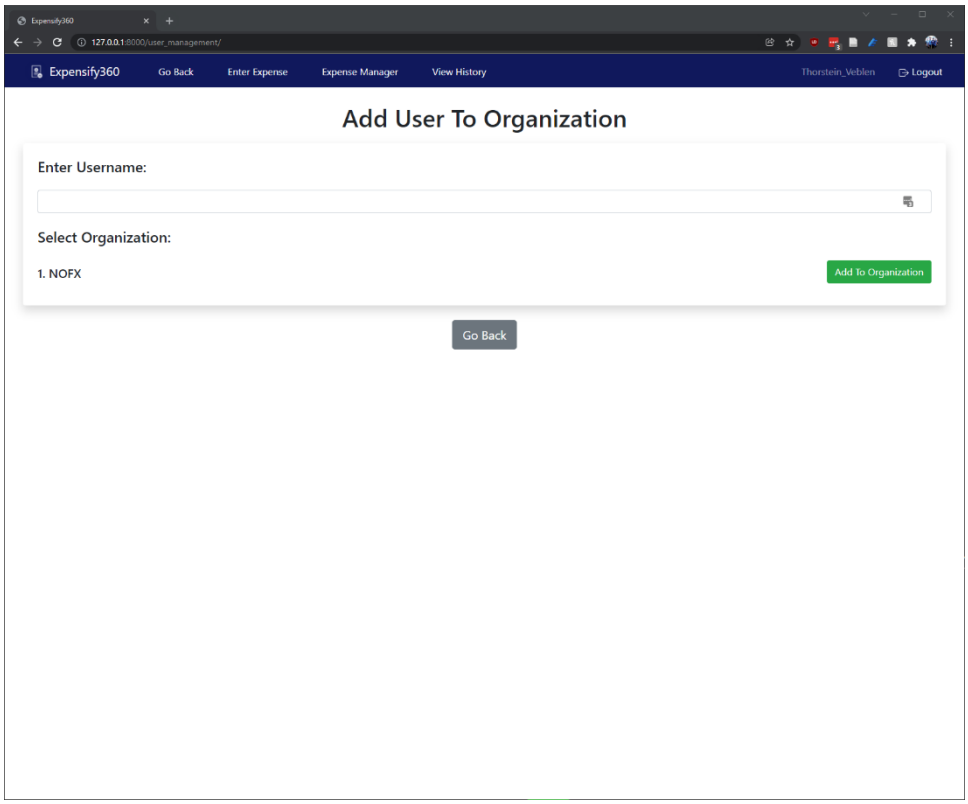


Figure 29 - Project User Management Screen

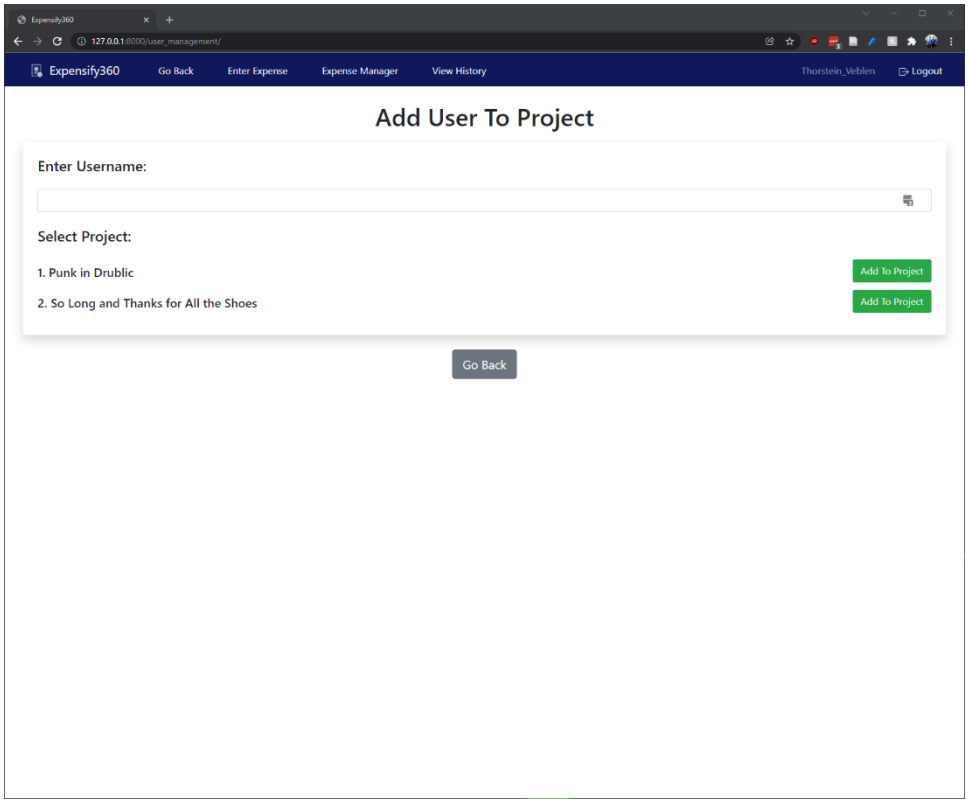


Figure 30 – Expense Management Screen

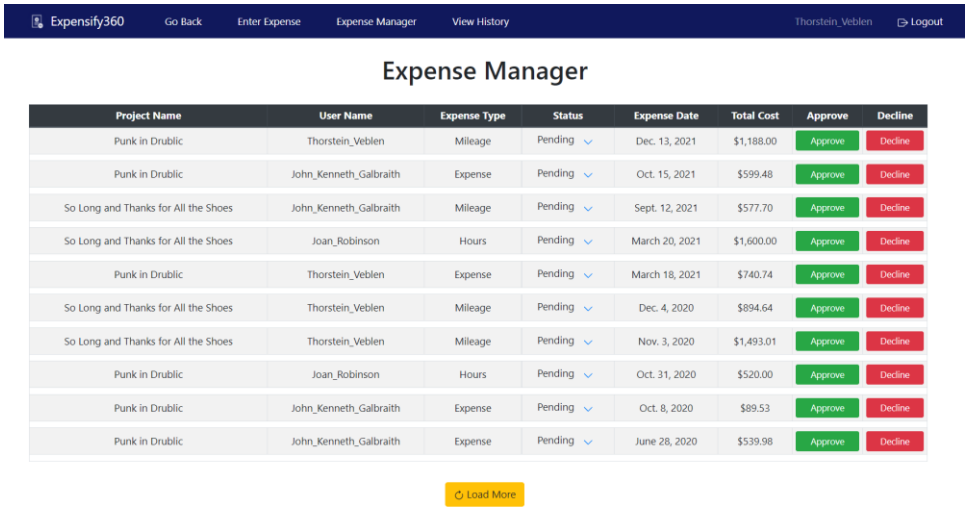
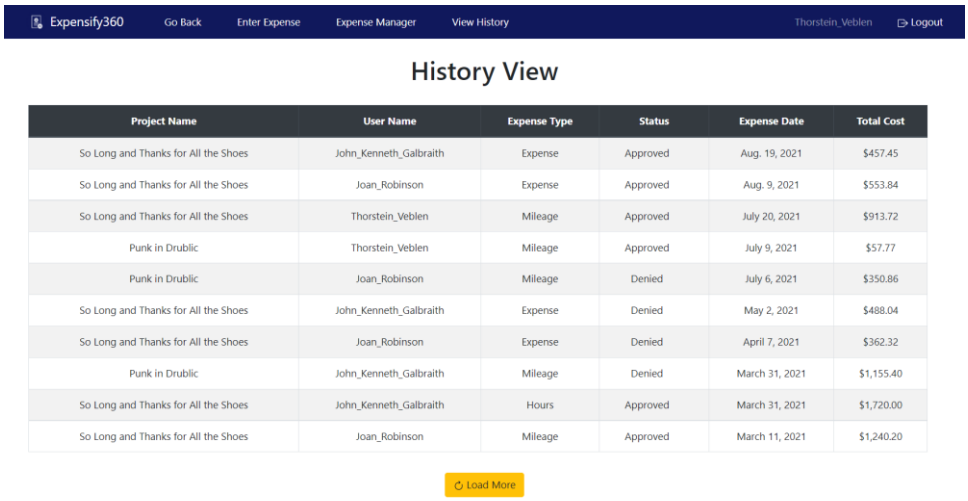


Figure 31 – Expense History Screen



4.2.4.2 Objects and Actions

The main user management page will render options as buttons. Create User will render a registration form similar to that shown on the registration page. Add User to Organization and Add User to Project buttons will render a form to enter a username and a button to add for each organization or project. Remove user will have a username field and a button to remove. All forms will display error messages on failure and a success message reiterating the action performed.

5 Restrictions, limitations, and constraints

- Latest versions of desktop browsers Chrome, Firefox, Microsoft Edge, Opera and Safari must be supported.
- This web application will not support Internet Explorer versions before version 11.
- For Android and iOS devices, the latest versions of Chrome, Firefox and Safari must be supported.
- Android version 6 or higher and Android System WebView will be required.
- Proxy browsers such as Opera Mini, Opera Mobile's Turbo mode, UC Browser Mini, Amazon Silk will not be supported.
- The information store will be an SQL database

6 Testing Issues

Test strategy and preliminary test case specification are presented in this section.

6.1 Classes of tests

- Performance Test – Load testing the application while it is running on a server by simulating the maximum use of the system.
- Accuracy Test – determine if queries return the expected results.

- User Interface Test – Have one external tester go through the pages that have the major functionality from the application.
- Accessibility Test - Ensure that users who are colorblind can navigate the application. Use Automated Accessibility Testing tool *axe DevTools* to check for serious issues on all pages.
- Security test – ensure that the app is secure from unauthorized users.
- Repeatability Test – Performing the same expense entry action multiple times should have the same result each time.
- Repeatability Test – Logging in and out of the application should not impact the values on the dashboard.
- Repeatability Test – Approving or denying expenses one after the other will be repeatable for all pending expenses.
- End-To-End Test – Validate the major functionalities of the application including Login, Register, Data Visualization, Form Entry Pages, Expense Approval and User, Project and Organization Management.
- User Interface Test – Users shall have different Expense type choices that are both visible and clickable.
- User Interface Test – The manager-level buttons are visible on the Dashboard can be clicked by the manager user.
- User Interface Test – An inexperienced user shall be able to click links from the Navigation Bar and navigate to correct Page from anywhere in the application.
- User Interface Test – The Load More button is clickable and visible if there are more than ten expenses.

6.2 Expected software response

- Performance Test – Response time for all requests should take between 60 and 30 seconds at a maximum.
- Performance Test – Dashboard graphs should not cause a loading delay to the main page.
- Performance Test – The database should be able to handle a minimum of 10,000 expense entries with no performance degradation.

- Accuracy Test – Returned chart data match input data.
- Accuracy Test – Expense entry total calculations return correct summed value.
- Accuracy Test – Reviewed expense files match originally uploaded copies.

- Security Test – All URLs are tested from an unauthenticated account and are redirected to the login page. Access to manager-only functionality denied to users not granted manager permissions.
- Security Test – The system prevents attempted SQL injections.
- Security Test – The system lets the user know they input the wrong password and won't sign them in.

- Repeatability Test – All duplicated queries have the same output result.
- Repeatability Test – Logging out and back into the application all forecast predictions remain the same.

- Repeatability Test – All expenses can be to be Approved or Denied repeatedly until there are no longer pending expenses.

- User Interface Test – The page loaded from any of the Expense type choices in the Expenses page will be correct according to their selection.
- User Interface Test – Clicking any of the Manager Dashboard buttons will not throw an error and will navigate to the expected page.
- User Interface Test – An external user that is unfamiliar with the application should be able to follow a checklist of instructions and navigate to the correct pages accordingly.
- User Interface Test – Users that click the Load More button below the Expense History or Expense Manager table will have the page display ten more rows of expenses.

- Accessibility Test – Colorblind user should be able to perform all primary app functions unimpeded. Automated Accessibility Testing tool should return no serious issues in any of the pages.

- End-To-End Testing – Users are successful in registering a new account, logging in with an account, entering a new expense to an expense form, having a manager-level user approve the item.
- End-To-End Testing – Manager-level users can create an organization, create a project, create and add users, delegate project management, review expenses and view expense history.

6.3 Performance bounds

- Must be compatible with the latest versions of Chrome, Firefox, Edge, Opera, and Safari
- Page loading should take a maximum of 20 seconds.
- The system should take no longer than 10 seconds to process a form.
- Dashboard data should populate in less than 20 seconds.
- Unauthenticated users are redirected to login screen.