**I-House App**

|  |
| --- |
| A Project Report Presented to  The Faculty of the Computer Engineering Department |
| San Jose State University In Partial Fulfillment Of the Requirements for the Degree Bachelor of Science in Computer/Software Engineering |

|  |
| --- |
| By |
| Altemush Bhatti,  Elias Habash,  Raj Makda,  Daanyaal Ahad Saeed |
| 08/2018 |

|  |
| --- |
| Copyright © 2018 |
| Altemush Bhatti, Elias Habash, Raj Makda, Daanyaal Ahad Saeed |
| ALL RIGHTS RESERVED |

|  |
| --- |
| **APPROVED FOR THE COLLEGE OF ENGINEERING** |
|  |
| Badari “Ishie” Eswar, Project Advisor |
|  |
| Professor Rod Fatoohi, Instructor |
|  |
| Dr. Xiao Su, Computer Engineering Department Chair |

**ABSTRACT**

**Developing a Housing and Residence Halls Management Application**

The International House of San Jose State University—I-House for short—boasts a welcoming and inclusive living environment with residents from diverse cultures and countries – as well as the United States of America. As with any residence hall, the International House has managerial staff that carries out tasks to make sure things run smoothly on a daily basis. This entails having highly organized operations, effective communication, a holistic, welcoming student community and the necessary tools to accomplish that.

Currently, the I-House office staff must resort to completing various tedious tasks by hand, that are otherwise achieved within seconds on a smartphone or computer. Furthermore, many other functions require the unnecessary use of paper, as opposed to using a digital and sustainable substitute. Last but not least, the office does not have an effective way to communicate with all the residents of the house except through Facebook. Many residents, who do not have a Facebook login, have to create an account in order to receive important alerts and updates.

Our project streamlines these processes in a single web application with features suited not only for the International House but also for any residence hall or housing management service. Not only will this service benefit the managerial staff, but also the residents as well as they will be aware of any events the I-House is holding through the events page. By creating this web application, we can centralize all operations, making them accessible, efficient, and paper free!

|  |
| --- |
| **Acknowledgments** |
| [Your Acknowledgement statements are presented here]. |

**Table of Contents**

**Chapter 1 Introduction**

* 1. Project Goals and Objectives
  2. Problem and Motivation
  3. Project Application and Impact
  4. Project Results and Deliverables
  5. Project Report Structure

**Chapter 2 Background and Related Work**

* 1. Background and Used Technologies
  2. Literature Survey
  3. State-of-the-art Summary

**Chapter 3 Project Requirements**

3.1 System (or Component) Functional Requirements

3.2 Non-Functional Requirements

3.3 Context and Interface Requirements

3.4 Technology and Resource Requirements

**Chapter 4 System Design**

4.1 Architecture Design

* 1. Interface and Component Design
  2. Structure and Logic Design
  3. Design Constraints, Problems, Trade-offs, and Solutions

**Chapter 5 System Implementation**

5.1 Implementation Overview

5.2 Implementation of Developed Solutions

5.3. Implementation Problems, Challenges, and Lessons Learned

**Chapter 6 Tools and Standards**

6.1. Tools Used

6.2. Standards

**Chapter 7 Testing and Experiment**

7.1 Testing and Experiment Scope

7.2 Testing and Experiment Approach

7.3 Testing and Experiment Results and Analysis

**Chapter 8 Conclusion and Future Work**

**References**

**Appendix**

Appendix 1 [Title] (for example, your selected hardware/software standard)

Appendix 2 [Title] (for example, your detailed algorithms for your solutions)

**List of Figures**

Figure 1. Login Sequence Page 21

Figure 2. Adding a New Event Page 21

Figure 3. View GIA Schedule Page 22

……

Figure M. Figure Title---------------------------------------------------------------------- Page No.

**List of Tables**

Table 1. Table Title Page No.

Table 2. Table Title Page No.

Table N. Table Title Page No.

[On the same line as each section heading, list the major contributors to that portion of the document.  Note, it is an academic integrity violation to list your name on a section if you were not a major contributor.  Similarly, it is an academic integrity violation if you are a major contributor but allow another group member to list his/her name unless (s)he was a significant contributor.]

[Update each of the following sections reflecting what was actually implemented. Ensure that it is written in the proper tense as a common mistake is just to copy the 195A report that is in the future tense, not the tense that reflects the end of 195B after the project has been completed. This requires some attention when you are updating the report in the middle of the semester and items are not quite complete yet.

While EACH section should be reviewed and updated, the following highlights in green note areas that have been changed from the 195A Report Format that may require particular attention. Highlight all changes in this report from 195A in yellow]

# Introduction

## 1.1 Project Goals and Objectives [Describe what are the goals and objectives of the project. In addition, it covers the context in which the project was placed.]

The goal of this project was to provide a fast, efficient, secure, and user-friendly web application for the I-House staff and residents. To accomplish this, we are offering a digital solution which increases day to day workflow productivity and eliminates the need for paper-use by I-House staff. The application also provides a tool for easy and effective communication between staff and residents.

## 1.2 Problem and Motivation [Describe the problem, motivation, and needs of your project. You need to address why this project is important and what is the problem you have addressed.]

The current problem is that the I-House staff does its management work using paper and pen, which is less efficient than using technology. Nowadays, technology is used in many aspects of life, including management of I-House operations. The app that we have built allows the I-House to use software technology as a means of managing and scheduling office work, as well as communication. Such functions will enable the office staff to work more efficiently without having to resort to using much paper. This also lets the office staff relay essential notifications and information which eliminates the need to use social media as a means of communication. When we approached the International house with our initial proposal, they too saw the benefits of our solutions and have worked with us since to set requirements and provide valuable feedback.

## 1.3 Project Application and Impact

[Describe the application of your project results, and its impacts on academic, industry, and society.]

Although the scope of our project is currently focused on providing a customized (unique) solution to the SJSU International House, the core concepts could extend to many other use cases. For starters, since our web application works in both theory and as a proof of concept, it could be offered to other residence halls, starting with the dormitories on the SJSU Campus. Of course, many of its current functionalities would have to be adjusted or altered to be relevant to each residence hall, but the foundation will remain.

Another potential impact that this application has on society is the formation of an International Student Network. If we were, in theory, ready to roll out our product in other countries, we could connect the users of our products to establish communications between students from different universities around the world. Our users, in this case, the students/residents, could share their experiences of studying in their respective countries.

## 1.4 Project Results and Deliverables

[Describe your actual project results (such as a system, and a component) and project deliverables (such as the report, prototype, code, etc.).]

This project is an application made for the I-House staff, who can utilize it to facilitate their work. Some key features of the app include a job scheduling module for scheduling the work shifts of the I-House staff, a work log for student assistants to fill. We have also added a user management feature which allows an administrator to monitor the accounts of the staff, an event scheduler that can be used to schedule different I-House events, and an application which enables residents to apply for I-House jobs. All the features added according to the needs of the office staff at the I-House are included in the final product. The project was completed in November 2018 along with an accompanying project report. The source code will be delivered with documentation that is easy to understand for future developers.

## 1.5 Project Report Structure

[Introduce the following sections of the document].

What follows is our project report and findings. In Chapter 2 we discuss the background of our project as well as any preliminary research we conducted. Chapter 3 outlines all the requirements of our project, from the domain and business requirements to any technological and resource requirements. We discuss our system’s design in Chapter 4, including constraints and trade-offs we have made along the way. Our implementation and advanced solutions are explained in Chapter 5. Chapter 6 goes over any tools we utilized and standards we adhered to while developing our application. Chapter 7 highlights the testing, the experimentation scope, our approach for the project, and the results of our project. Finally, in Chapter 8 we conclude our project and discuss future work.

# Chapter 2 Background and Related Work

## 2.1 Background and Used Technologies

[Provide the necessary background of this project, including concepts and knowledge (e.g design patterns, asynchronous programming, project estimation, scientific and mathematical theories), along with technologies (e.g. PHP, MySql). In addition, provide an updated table of courses you have taken that you applied to the project and how you applied them.]

The web application is using a 3-Tier architecture which is being implemented using the MERN stack. We are using React.js and Redux for the front end, Node.js and Express.js for the backend, and MongoDB for data persistence. Using the MERN stack means that JavaScript will be our primary programming language which follows an asynchronous programming pattern. React.js will help build sophisticated user interfaces easily because it allows us to write reusable components. Node.js will give us the operating system independence as it will run on Windows, OS X, and Linux, for development purposes. It is supported by a vast module library which makes it great for integrating other technologies (Pal, 2016). Express.js is a framework within Node which will be used to set up a server to communicate with the database and provide the data as JSON objects through a REST API. React.js will connect with this API to grab application data, process it and present the components to the user. Using Javascript Object Notation (JSON) as the format for exchanging data on all the layers removes the need to use libraries to parse the data during the client-side and server-side interaction. MongoDB is a great choice to use as a database because being a NoSQL database it is very flexible. All the technologies in the MERN stack are open source, so we have the benefit of new developments and updates that are added by developers in the community.

This technology stack presented a learning curve for our team, and we have absorbed a lot of knowledge while building the application. We have utilized Udemy.com and StackOverflow for video tutorials and code snippets which are great resources. We have used Agile methodologies to track the development and GitHub has been used to manage the source code and bugs.

## 2.2 Literature Search

[Similarly, present your updated literature search adding to those that you explained in Chapter 1 of 195A workbook.]

The aim of the International House Application is improving workplace efficiency. Chen et al. [1] mentioned in their paper that Student Dormitory management systems (SDMS) - similar to ours - can be based on a 3-layer architecture & web server. The three layers are Presentation, Business Logic, and Data Access. The presentation layer is the front-end with all the interactive features. It interprets information that comes from the business and data access layer and presents it to the users. The business logic layer is where all the functions take place, such as authentication, time scheduling, roommate matching, etc. The data access layer comprises of the database and data manipulation operations, such “as adding, deleting, updating and searching” (Chen, Z. F. et al., 2014). This type of architecture allows for greater collaboration and makes the division of tasks efficient. The main focus of our development is in the business logic layer as it is the area that will make or break the web application. Furthermore, the 3-layer architecture also allows for a decreased maintenance cost and workload for managers.

In yet another paper, X. Zhang et al. discusses a university dormitory management system which they have built on an Agile development architecture [3]. By learning from their development process, it assisted us in creating effective project completion strategies. Although their architecture is based on the older .NET platform, and thus many technologies associated with it are outdated, it mentions how agile development is an excellent opportunity for team members to improve their technologies and the levels of cooperation. The Agile process is an “iterative development model” with “incremental deliveries” (X. Zhang et al., 2011). It divides the projects into smaller and manageable sub-projects each of which should take 1 to 6 week(s) to complete. The main focus of each of these subdivisions is to maximize performability and allow for seamless and continuous integration. The agile process also accommodates the changing demands of the customer.

We decided to proceed with our project by adopting a DevOps strategy. DevOps, short for development and operations. As Sharma and Coyne [2] describe it: In broad terms, DevOps is an approach based on lean and agile principles in which business owners and the development, operations, and quality assurance departments collaborate to deliver software in a continuous manner that enables the business to more quickly seize market opportunities and reduce the time to include customer feedback. (Sharma & Coyne, 2017, p. 1) This allows for development and quality assurance (QA) teams to comprehend how the application performs and behaves well before deployment (Sharma & Coyne, 2017, p. 6).

Furthermore, with its practice of continuous releases, new features are released and introduced to customers and users whenever possible. The project development operations would be split into four environments: Development, Test, Stage, and Production. Each context requires its own form of software quality testing, ensuring that the final build is stable and ready for deployment. Due to our project’s nature, the fact that we have a customer, and many potential users, DevOps is the ideal strategy to execute. Using DevOps, we can get a working product out quickly, while still adding and improving features from the constant feedback.

## 2.3 State-of-the-art Summary

[A smaller, one-page summary follows the literature review. Please refer to ‘State-of-the-Art Summary’ section in Chapter 1 of 195A workbook. You should provide an updated state-of-the-art summary here.]

Currently there are many problems in the SJSU International House. However, we have built an all-in-one one-stop remedy. We wanted to make sure our project is an efficient tool to solve these problems. This is why we used all leading-edge tools and techniques, such as React, Node and MongoDB, available for the area our project is focused on.

The very first problem we wanted to overcome is the availability of the system on maximum possible platforms to achieve a maximum user base. Thus we have built our application with React, a popular JavaScript library for creating user interfaces. Our web platform and user interface will be developed using React. As previously stated, using React means that we do not need to go into Operating system specific programming languages like Java or Swift to build different applications for iOS, Android and the Web.

The reason we picked MongoDB over other popular databases is that it provides a cloud-based solution with mLab for all storing data which allows seamless syncing with all platform application. It will enable easy data access by implementing RESTful HTTP API compared to the JSON format. The system architecture depends upon the use of RESTful API and using MongoDB provides just that. MongoDB supports the use of JavaScript and Node which is also used for implementing the server side for both applications. MongoDB also addresses the security concerns related to databases. Finally, it provides a flexible data model with reliable performance making it a perfect fit to meet the database requirements for this application [1].

As the project advanced and we started to explore improvements in other possible areas of development, integration of additional technologies would be researched and noted.

# Chapter 3 Project Requirements

## 3.1

## System (or Component) Functional Requirements

### 3.1.1 Authentication and Authorization

*3.1.1.1.* A user should be able to successfully log in with an email address and password

*3.1.1.2.* A user should be shown the correct error message if they log in with invalid credentials

*3.1.1.2.1.* No user found error if they log in without signing up

*3.1.1.2.2.* Invalid password error if they log in with an incorrect password

*3.1.1.3.* A user should be able to register with their Sjsu Id, First name, last name, email address and a password

*3.1.1.4.* There shall be 4 authorization levels for users - admin, RA, GIA employee and resident

*3.1.1.4.1.* An admin user should be able to elevate the authorization level for a user

*3.1.1.4.2.* An admin user should be able to delete the account for a user

*3.1.1.4.3.* An admin user should be able to view a list of all user accounts in the system

*3.1.1.4.4.* A RA user should be able to create, update and delete events

*3.1.1.4.5.* A RA user should be able to submit a timesheet

*3.1.1.4.6.* A GIA user should be able to submit a timesheet

### 3.1.2 Timesheets

*3.1.2.1.* Only GIA and RA (student employee) users should be able to submit their timesheet for a single time period

*3.1.2.2.* Each month should have two time periods

*3.1.2.2.1.* 1st-15th

*3.1.2.2.2.* 15th-end of month

*3.1.2.3.* There should be four inputs for each job done in a particular time period

*3.1.2.3.1.* Date of job done

*3.1.2.3.2.* Description of job done

*3.1.2.3.3.* Time In and Time out

*3.1.2.3.4.* Total Hours worked on the job

*3.1.2.4.* The time period for a timesheet should be automatically set depending on the current date

*3.1.2.5.* The SJSU id of the user submitting a timesheet should be automatically set for the logged in user

*3.1.2.6.* A user must be logged in to submit a timesheet

*3.1.2.7.* A user should be able to view all the timesheets submitted by themselves

*3.1.2.8.* An admin user should be able to view all timesheets submitted by selecting a time period

*3.1.2.9.* Timesheets should have four different statuses

*3.1.2.9.1.* Approved

*3.1.2.9.2.* Rejected

*3.1.2.9.3.* Pending

*3.1.2.9.4.* More Information Required

*3.1.2.10.* An admin user must be able to approve, reject, or request more information on a particular timesheet

*3.1.2.11.* The student employee user must be able to view the status of their submitted timesheet

*3.1.2.12.* A rejected or approved timesheet cannot be resubmitted by the user

*3.1.2.13.* A more information timesheet can be resubmitted by the user

### 3.1.3 Events

*3.1.3.1.* An admin or RA user should be able to create a new event

*3.1.3.2.* A new event should only be created if it is taking place in the future

*3.1.3.3.* An admin or RA user should be able to update an upcoming event

*3.1.3.4.* An admin or RA user should be able to delete an upcoming event

*3.1.3.5.* An event should have title, date, location, description, organizer and an image

*3.1.3.6.* All upcoming events should be displayed on the home page

*3.1.3.7.* All upcoming events should be viewable by everyone who visits the website

*3.1.3.8.* Events should automatically expire 24 hour after the day they took place

### 3.1.4 Scheduling

*3.1.4.1.* An admin user should be able to upload all GIA employee application data using a csv or xlsx file

*3.1.4.2.* An admin user should be able to view the availability of all users who have submitted a GIA application

*3.1.4.3.* An admin user should be able to use the availability to set a schedule for the GIA employees

*3.1.4.4.* The schedule should be set for a particular semester

*3.1.4.5.* Every GIA employee should receive a notification for their schedule

*3.1.4.6.* The schedule should be set for a single week repeated throughout the semester

*3.1.4.7.* The schedule should have 5 slots for any given day

*3.1.4.8.* The slots for the schedule should be 10-12pm, 12-2pm, 2-4pm, 4-6pm and 6-11pm

*3.1.4.9.* Schedule for Saturday and Sunday should only require the 6-11pm slot

## 3.2 Non-functional Requirements

### 3.2.1 Availability

*3.2.1.1* Application needs to be up and running at all times with a minimum downtime of 1 hour per month for maintenance.

### 3.2.2 Usability

*3.2.2.1.* Users should be able to easily create a user account in 2 minutes.

*3.2.2.2.* Navigation between views is labeled with clear workflows.

*3.2.2.3.* Buttons are either labeled clearly or used with an icon and clearly identify the given function.

### 3.2.4 Reliability

*3.2.4.1.* The application should inform the user when the wrong data is inputted and should ask for resubmission of input.

*3.2.4.2.* The application should encrypt all data during communication and notify the user is the communication is not secure.

*3.2.4.3.* The application should be resilient to unexpected exceptions.

### 3.2.5 Performance

*3.2.5.1.* The application should support all 75-80 simultaneous users of the I-House (residents and I-House staff members) without slowing down.

*3.2.5.2.* Under a normal load, the screen should load within one seconds

### 3.2.6 Supportability

*3.2.6.1.* The application should run on all phone platforms and all computer devices.

*3.2.6.2.* The application should be supported by Chrome, Firefox, Safari and Edge

### 3.2.7 Packaging

*3.2.7.1.* The application should be built and run on the I-House servers.

## 3.3 Context and Interface Requirements

[Specify the context environments supporting your development, testing, and deployment

of your project results. You also need to describe the interface requirements for your

hardware/software components and system.]

There should be two environments for the development of the application: Staging (Development) environment and a final Production environment. The GitHub repository should be set up with a continuous integration pipeline that will deploy a stable ‘develop’ branch to the staging environment and the ‘master’ branch to the final production environment. The CI/CD pipeline will run all unit and integration tests with newly checked in code before allowing deployment to the staging or production environments. Our pipeline will also require code reviews from other developers on the team in order to maintain code style and standards.

The staging environment should be a server on AWS or Google Cloud platform with access only to developers through an SSH tunnel. The production environment should be a server in the International house server room which will provide access to the application only on the network inside the International house. For deployment a Unix server is required with Node JS (version 8) and MongoDB installed.

## 3.4 Technology and Resource Requirements

[List the requirements for hardware (devices, components, systems, etc.) and software (compiler, database, middleware, etc.), technologies. This section must include textual description accompanied with tables.]

|  |  |  |
| --- | --- | --- |
| Client Side | React.js | - The client should be supported by Chrome, Safari, Firefox, and Edge  - The ES2015 code should be transpiled to vanilla JavaScript code that can be understood by the Browser  - The code should follow Airbnb linting style guides |
| Server Side | Node JS / Express JS | - The backend should be secured by authentication tokens  - The backend should have middleware that verifies the API Key for every request  - The backend should provide CRUD operations on the data models |
| Database | MongoDB | - The database should be hosted on the deployment server  - The database should have admin privileges  - The database should be secured by authentication credentials |

### React.js

React.js is a JavaScript library from Facebook to build complex User Interfaces. It is the primary library that was used to render UI Components onto the web pages for our project. The UI was divided into small, generic components that can be used as reusable templates throughout the app.

### Node.js

Node.js was used to build the server-side for the application and manage all the data coming in from the client side and database. A REST API was created for the different entities in the system to make it easier to manage the database. The REST API was secured by authentication tokens to prevent unauthorized access to the database.

### MongoDB

MongoDB was the central database that stored all the data required for the application. The production database was separate from the development database so that data is not accidentally lost or corrupted in the development process.

### GitHub

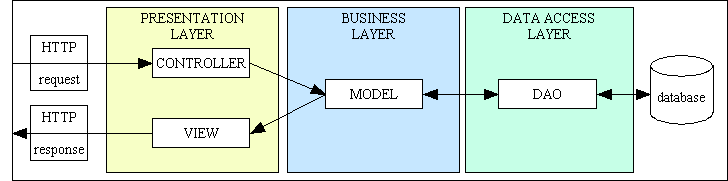
Github was the primary tool that was used to manage all the source code. It is the most widely used version control system out there and also a critical tool to learn as developers. The repository was used to set up a continuous integration pipeline to fulfill the deployment and testing requirements.

# Chapter 4 System Design

## 4.1 Architecture Design

The primary purpose of this section is to provide an architectural overview of the university management system developed for the International House at SJSU. The architecture of this software system defines the main components as well as the structure of the project, and the means of communications between the different layers of the application.

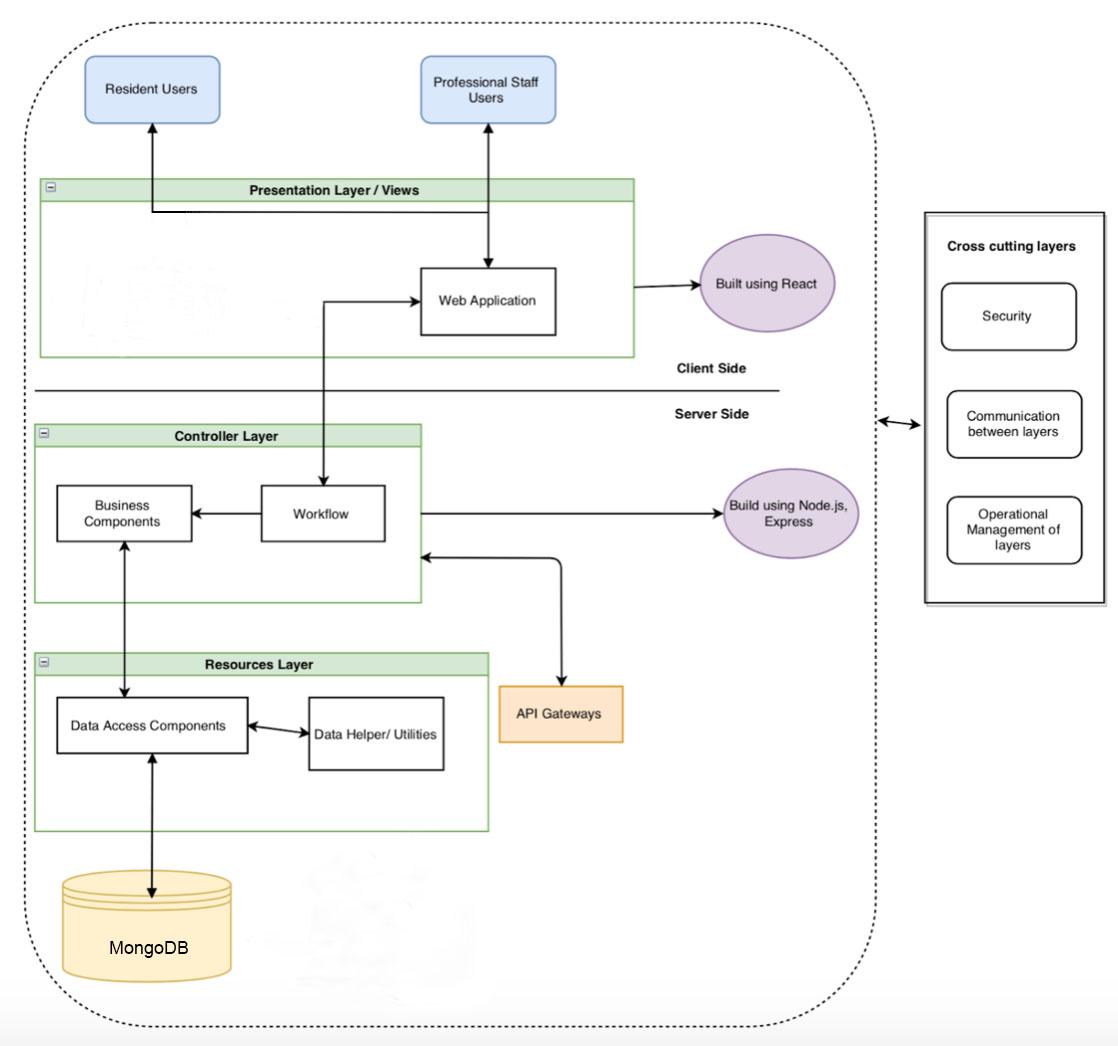
The system utilized a 3-Tier Architecture as the blueprint, with each layer having its own components with unique design patterns. The figure below shows the three main layers of the 3 Tier architectural pattern: Presentation layer, Business layer, and Data access layer with their individual components. This high-level structure for the application further elucidates how different parts will interact with each other. The reason for selecting a 3-Tier architectural pattern was to maintain a separation of various aspects of the app for simplicity and prompted us to use modular code, which was easier to manage. Another benefit of the 3 Tier Architecture was that changes in business rules in the future can be easily integrated without breaking changes to the UI or Database modules. It also provides the flexibility to easily update to a newer tech stack for for each layer independently (Kapahi, 2017).



## 4.2 Interface and Component Design

A little description of the three layers and the technology stack used to implement them is given below:

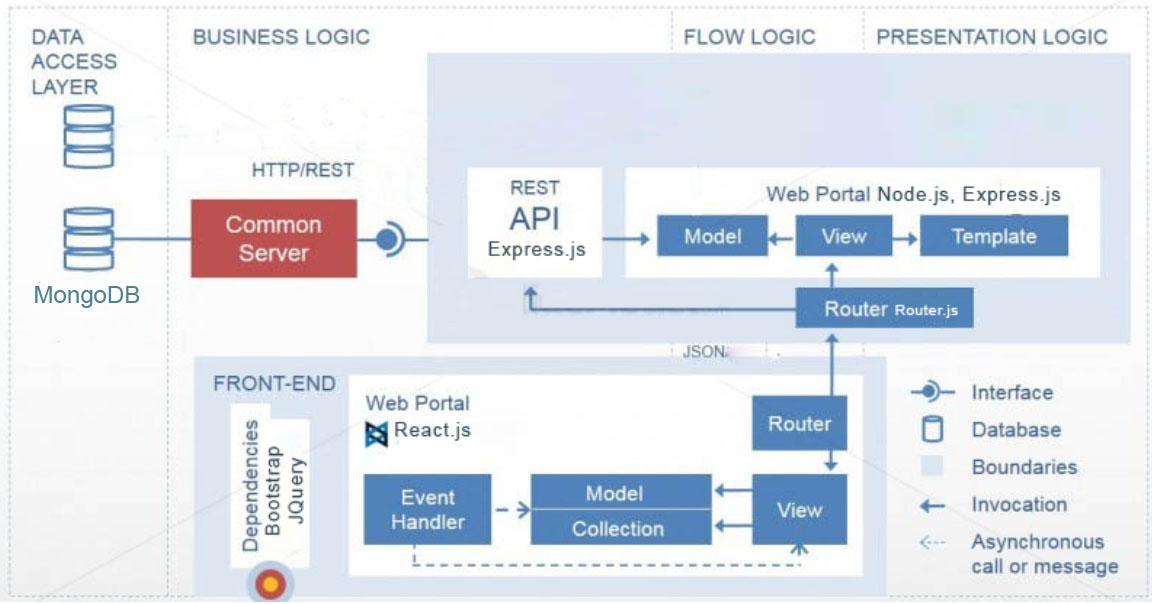
* **Presentation layer:** The presentation layer will be viewed by the users using their mobile or computer devices for interacting with the system. This layer will be built using an open-source front-end framework called React.js and will be dependent on Bootstrap 4 and react-bootstrap libraries. Reusable components that were used across the system were developed. Each component was part of a UI page(s) and maintained its own state. The data to be displayed was fetched by making http requests to a backend server, which handled all the business logic. This layer consists of an event driven architecture where a user interaction drives an action handled by the business layer.
* **Business layer:** The business layer handled requests coming from the presentation layer and sent JSON data back as a response. All the central business rules and logic that drives the system are also implemented in this layer. This should make it easier to update the application if there are changes in the business logic. A server to handle requests from the user was developed on Node.js. Routing the user’s requests to correct phase of the workflow was done in Express.js. An database driver connected to the database sent requested data back to the user through http responses. This layer also provided additional services such as authentication and user management using a RESTful API.
* **Data access layer:** The data access layer stored all the information and data for the application. It connected MongoDB was used because it provides a simple way to structure the data models and flexibility in adding new attributes. ("Using a Three-Tier Architecture Model")



## 4.3 Structure and Logic Design

[Present the detailed structure and logic design for your hardware/software components and processes. This section must include textual description accompanied with diagrams. If scientific or mathematical fundamentals are used for your project algorithm, specify what kind of formula or theory has been applied.]

The figure below depicts communication between layers using messages and application programming interfaces (APIs). The interaction between the data access layer and the business layer is through an open source API called mongoose.js for Node.js. Mongoose acts as a driver to connect, query, and bring data back to the database through HTTP requests using asynchronous callback functions in the JSON format. A REST API was implemented in Express.js that manipulates data through CRUD (Create, Read, Update, Delete) operations. The API also acts as a middleware that contains the business logic and routes the request to the correct workflow. The production build for the front-end includes the dynamic views that are served as static files using this application server.



## 4.4 Design Constraints, Problems, Trade-offs, and Solutions

### 4.4.1 Design Constraints and Challenges

[Present your design constraints in different perspectives, such as economic, resources, society and environment, hardware/software, mathematical/scientific theories and safety and reliability.]

***4.4.1.1. Resources***

There are several different resources this project required. Our first concern was time management. When considering the scope of our project, efficient utilization of time was factored in. The number of features we wanted to implement has also been influenced due to the lack of human resources. Having more people working on additional components of the application would have sped up development and allowed for the inclusion of more features.

***4.4.1.2. Social***

Considering that this software is being developed for the SJSU International House, there are many social aspects that we needed to discuss. Our biggest challenge in this aspect stems from the fact that most residents are non-native English speakers. Furthermore, we wanted to reach a wide audience of users.

***4.4.1.3. Software***

When we first analyzed the challenges set by undertaking this project, we tried to consider through what/which platforms we would reach our users. The choice was between creating native phone applications or a web application. Due to the resource as mentioned earlier constraints, we decided to proceed with the development of the latter. We also had to make sure our user experience is consistent across all web browsers.

***4.4.1.4. Reliability***

We understand that our software is going to be used quite frequently in the International House; therefore reliability is of great concern. So much so that it takes precedence over the implementation of certain features.

***4.4.1.5. Sustainability***

Now that we have completed our Senior Design Project, the team is graduating and moving on to other affairs. One of our concerns is regarding the continuation of our project post-completion. Much thought has been put into this dilemma and arrangements have been proposed.

### 4.4.2 Design Solutions and Trade-offs

[Document your approaches to cope with the given constraints. Present your design trade-off decisions and solution selections to deal with these constraints and problems and challenges.]

To tackle the constraints highlighted in the previous section, we have documented the following approaches:

***4.4.2.1. Resources***

To solve both of our resource challenges, time and manpower, we had to reconsider what we had initially planned for the project. We had to cut down our feature list from 15 features to only 7. Now although this severely limits the functionality of the I-House App, the highly desired requirements are delivered.

***4.4.2.2. Social***

Since we do not have any language experts in our team, we had to proceed in coding the app with one language in mind. Fortunately, there is no considerable use of the English language. In other words, there are no long paragraphs that require users to read to operate or navigate the application. This allows users with a fundamental mastery of the English Language to run it with ease.

***4.4.2.3. Software***

As we decided to take the approach of building a web app, the project was developed using the MERN stack. The MERN stack allowed the application to be available for use on all platforms, through their web browsers (with a few noticeable exceptions, i.e. Internet Explorer). There are little to no trade-offs when it comes to developing a web application over a native app, except for performance. However, since we are not trying to mimic a native phone application and performance is already at a desirable level, we are satisfied with our design choice.

***4.4.2.4. Reliability***

To increase the reliability of our application, we needed to ensure that it was well tested. The trade-off comes in the form of once again reducing the number of initially planned features. Furthermore, we had to reduce the customizability of the I-House App concerning what the I-House staff will be able to alter. For example, being able to change the color schemes of the app based on personal preference. By reducing the customizability of the project, we reduce the number of corner cases that we could not have accounted. Of course, this challenge could have been overcome if we spent more time testing our app with I-House users.

***4.4.2.5. Sustainability***

The sustainability of the I-House App after the conclusion of this course and post-graduation will depend on our Project Manager as well as the IT team at the International House. Extensive documentation has been written, so that future developers understand how the project has been set up. This documentation will also outline the features we did not have time to code but exist in the project backlog.

# Chapter 5 System Implementation

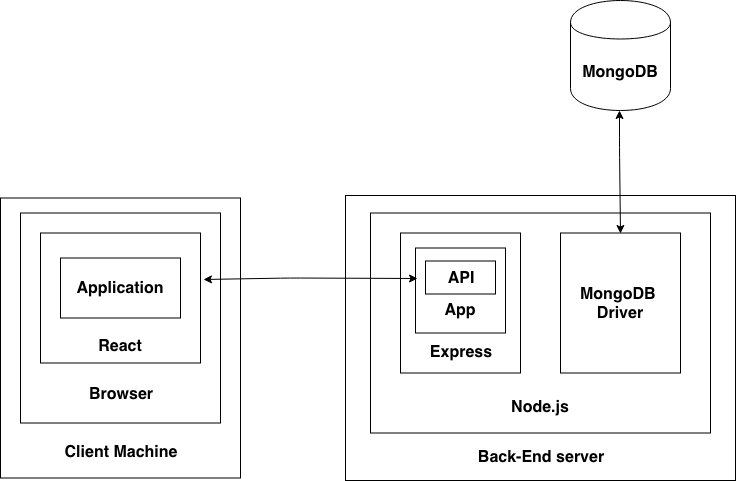
## 5.1 Implementation Overview

The implementation scope was focused on building a web application for the front end and a web server at the backend. For the front end React.js and Redux along with HTML and CSS was used for development. Styling of the components was done using Bootstrap and depended on the react-bootstrap library which provided out of the box React components. The back end too was implemented in Javascript which was dependent on Node JS as the runtime environment. The choice of database was MongoDB which was hosted on the cloud and connected to the backend using the MongoDB driver for Node JS. There were numerous other libraries and dependencies such as bcrypt to handle password hashing, JSON web tokens for web security, concurrently for task automation, and react-router for url mapping on front end. Together these technologies made up the MERN (Mongo, Express, React, Node) stack for application development. A continuous integration pipeline was set up in the GitHub repository using Travis CI. AWS EC2 servers were utilized for the staging (Development) environment. This was a high level overview of the implementation and the next section describes the implementation of each part of the stack in more detail.

## 5.2 Implementation of Developed Solutions

For our solutions to the problems of the I-House, we have built a scheduling module for the scheduling of jobs in the I-House. We have also created a Work log module to help with keeping track of workers’ jobs and hours worked. For both these modules, we utilized the libraries of React.js and Redux for building the components and keeping track of the application state. The use of React.js made it easy to create reusable components that allowed us to display them in multiple pages, while the use of Redux to manage the application state made it easy for the components to display parts of the application state.

Since we implemented the project using the MERN stack, we used JavaScript as the primary language for the backend and the frontend, which made it more accessible and required less from our part to use another language.



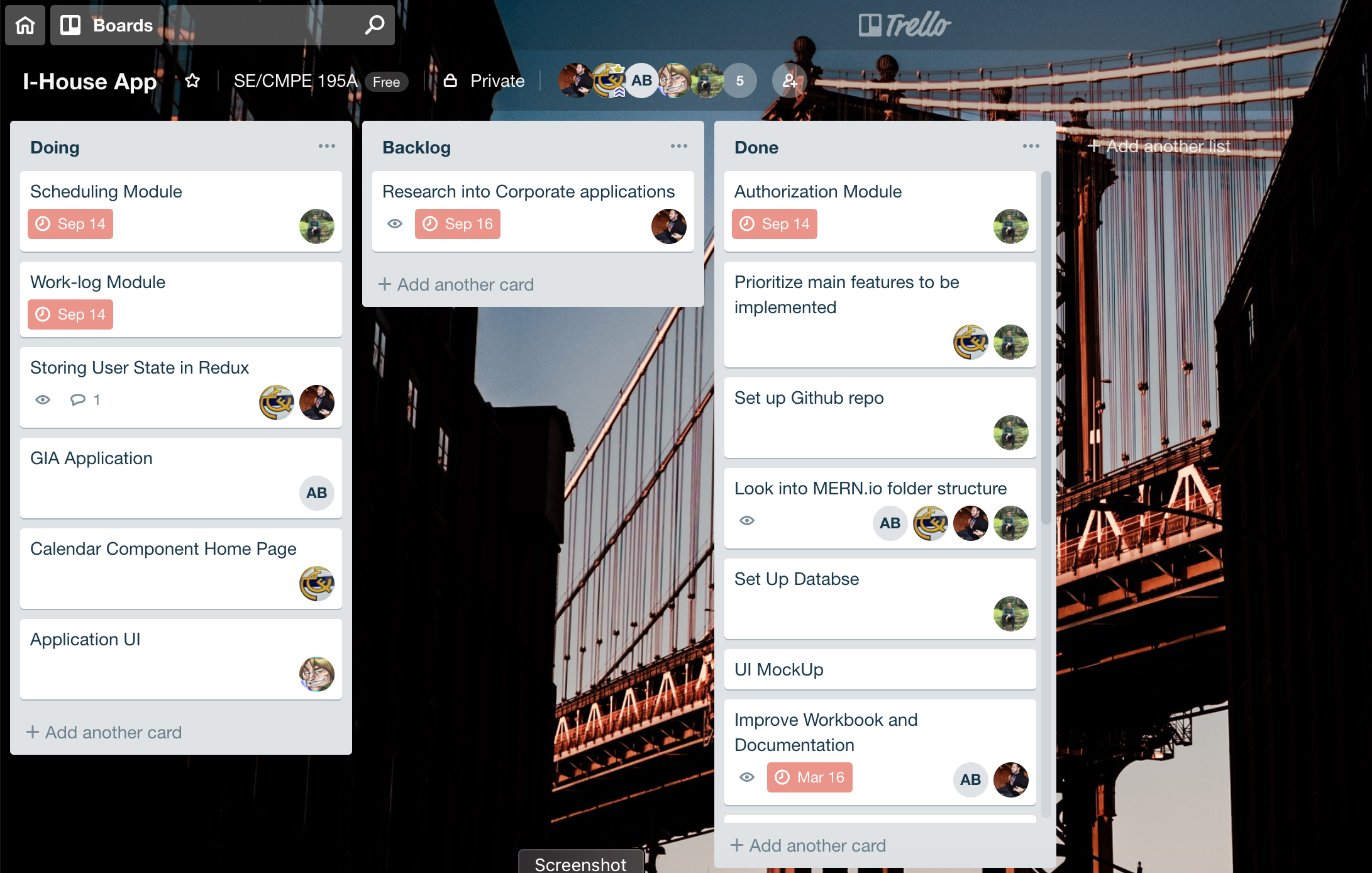
## 5.3 Implementation Problems, Challenges, and Lesson Learned

The major implementation problems and challenges were concerned with the front end. None of the team members had any prior working knowledge of React.js and Redux so there was a knowledge gap to begin with.. There were many resources available on the internet in the form of video tutorials and blogs that helped us get on the right track, and constant communication from our team to help each other also made a big difference in learning. Another challenge that the team faced was with setting up a continuous integration pipeline that automatically deployed production ready code directly to the I-House internal server. This was due to restricted access to the I-House server without the presence of their IT admin. The team overcame this difficulty by doing manual deployments to the server each time there was a new release. A third challenge was ensuring the reliability and performance of the application when it would be deployed to production environment. This happened due to lack of experience writing performance tests for applications. Overall, the team anticipated many such big and small challenges throughout the process and learnt a ton through the process of resolving the challenges.

# Chapter 6 Tools and Standards

## 6.1. Tools Used

The tools used in the I-House App included various text editors, for editing the project code, Github for version control, and npm for running and developing the code. Text editors, such as VSCode and Atom, were easy to work with when writing code because they include code-specific syntax highlighting and organizing the code base using these tools was easy. Github was an efficient tool for committing different parts of the project so that no developer’s code would overlap. Github also helped in the organization of the project code. The npm package manager was used to run a development server to emulate what the project would look like running on a webpage and in the browser. Also, the team used Jest to help test all javascript code, which is a tool that is especially useful for React applications. Trello was used for project management and assigning tasks to each team member. The figure below displays the team’s Trello board which shows the collaboration of each team member in the project.



## 6.2. Standards

For our app, the team had several standards that we used. Most of the standards of our app came from the Reactjs website since React was a big part of our project. The website states all the rules and guidelines of React that we were following. We also followed the React design principles that can be found on the site. These principles include having optimized and stable code and having proper composition so that individual components don’t change others. For example, the composition standard states that components written by different developers must work with each other efficiently, and adding functionality to a component must not cause rippling changes throughout the codebase (“Design Principles”). React also highly values interoperability which allows React to work with other UI libraries, and we made sure we correctly implemented React when working with the UI of our app. A standard for our app’s code is standardization, which made our code look better and neat. This allowed us to be more organized and made debugging the system much more comfortable.

The team used Github for project management and for committing different sections of our app. For this, we had to follow git standards for different git commands such as branching and pull requests. These git standards allowed us to stay organized and have proper version control of our app without having any overlaps or other similar problems. Some agile methodology standards were used, as the team conducted sprint plannings, creations of sprint backlogs, weekly scrum meetings, discussed next sprint plannings, and retrospectives of each sprint. The team also adhered to using some of the IEEE and ISO standards, such as the IEEE 29119 standard for testing and ISO 9000 set of standards for maintaining the quality of the app.

# 

# Chapter 7 Testing and Experiment

## 7.1 Testing and Experiment Scope

Carrying out testing ensures our code behaves the way we want it to, or to rectify it if it does not — our testing process both manual and automated testing. Manual testing is carried out to experiment with the I-House App’s user interface and user experience. Since the user interface is something very subjective, there is a blurry line between what is right or wrong. The idea is to make sure that the user interface is intuitive to use and navigate. One way we are ensuring such results is by asking potential users to use our application at different stages of productions. Automated testing allows us to run a suite of tests, that is executed without any intervention. We believe the following types of testing provide sufficient screening for this project:

1. Unit Testing
2. Functional Testing
3. Usability Testing
4. Regression Testing
5. Performance Testing

## 7.2 Testing and Experiment Approach

**Unit Testing**

Unit Testing was performed for the smallest portions/units of the codebase. Each team member was responsible for applying unit tests for the code they wrote. The primary objective for this kind of testing was to ensure the correctness of code written by each team member. Jest, a popular Javascript testing framework, was used for running most of the unit tests.

**Functional Testing**

Functional testing is a kind of testing which verifies that every function performs in the manner as described in the functional requirement specification. It is a form of black box testing which is independent of the source code. For the I-House App, this testing was performed for the backend APIs and User Interface by manual processes. The system was given various inputs, and the output was checked against an expected output with a goal to make sure the system is functionally working as expected.

Worklog

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Scenario | Test Steps | Input | Expected Output | Actual Output | Pass/Fail |
| WT\_01 | Check to see if the total number of hours is automatically calculated correctly | 1. Enter the description of work completed 2. Enter # of hours for work 3. Repeat for multiple entries | 4, 4, 5, 8, 7 | 28 | 28 | Pass |
| WT\_02 | Check if the user can add more entry rows | 1. Click green ‘+’ button | Button click | New row to enter worklog information | N/A | Fail |
| WT\_03 | Verify worklog can be submitted | 1. Enter worklog information 2. Click Submit | Work completed | Alert notifying the user of the successful submission | “Your worklog has been submitted” | Pass |
| WT\_04 | Verify the correct time period is selected | 1. Navigate to Worklog page | N/A | The current Time period is displayed and selected | Time Period: 1- Jan 1st -15th | Fail |

Scheduling Module

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Scenario | Test Steps | Input | Expected Output | Actual Output | Pass/Fail |
| ST\_01 | Verify if all accepted Office Aides/Evening Receptionists are listed | 1. Navigate to Schedule 2. Select Office Aide or Evening receptionist | N/A | John  Abbey  Christy  Derek | John  Abbey  Christy | Fail |
| ST\_02 | Verify Schedule can be created and submitted | 1. Select members for each shift | Selecting names | Alert notifying the user of the successful creation of the schedule | “Schedule for Office Aides has been created/ updated” | Pass |
| ST\_03 | Verify Student Assistants receiving the schedule | 1. Create a schedule from an admin account 2. Navigate to Schedule in applicable resident account | N/A | Recently generated schedule | No/old schedule | Fail |

Student Assistant Application

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Scenario | Test Steps | Input | Expected Output | Actual Output | Pass/Fail |
| SAAT\_01 | Verify Student Assistant Application can be submitted | 1. Navigate to Application 2. Fill out the application | Valid entries in all input fields | Alert notifying the user of a successful submission of the application | “Thank you for submitting your application. We will get back to you shortly on your results” | Pass |
| SAAT\_02 | Verify Student Assistant Application is submitted | 1. Navigate to Application from an admin account | N/A | David  Yoko  Arjun | David  Arjun | Fail |

**Usability Testing**

Usability Testing is the process of verifying the ease with which a user can learn to operate the system. The target audience will consist of the staff and students of the international house at SJSU. When the demo is ready, a small set of the students and employees at the International House will be given the application to use and expose any defects. The primary goal of this type of testing is to verify the effectiveness and efficiency of the engineering of the application concerning the usage by the final user. It should be able to validate the usability requirements listed in the non-functional specification.

**Regression Testing**

Regression testing ensures the integrity of our application is maintained by continuously testing the application whenever new code is merged into the master branch. A continuous integration pipeline was set up using Jenkins to ensure that every time new code is checked in, all regressions tests are automatically run. The regression test suite consists of a partial set of the test cases to ensure a new feature does not break the previously implemented features.

**Performance Testing**

Performance Testing is a kind of testing which ensures that the application will perform under all types of workloads and stress. The most crucial test in under this is the load time of the website. It will be conducted at the end of every regression test by deploying the application to the International House servers and monitoring the load time in its production environment. A testing tool such as LoadView Testing or HP LoadRunner will be used to run the suite of test cases.

## 7.3 Testing and Experiment Results and Analysis

[Describe testing and experiment results and analysis. For example, test execution and test result summary, performance test result analysis, test coverage, bug distribution report, and so on. This section must include textual description accompanied with figures and/or tables.]

|  |  |
| --- | --- |
| ID | Current Status |
| WT\_01 | Pass |
| WT\_02 | Pass |
| WT\_03 | Pass |
| WT\_04 | Pass |
| ST\_01 | Pass |
| ST\_02 | Pass |
| ST\_03 | Pass |
| SAAT\_01 | Pass |
| SAAT\_02 | Pass |

# Chapter 8 Conclusion and Future Work

The SJSU International House is a cultural mosaic, highlighting the various traditions, customs, and values that residents bring with them from their homes. The idea of our project was to streamline the process in a single web application with features suited for not only the International House but also for any residence hall or housing management services. By creating this web application, it centralizes all operations making them accessible, efficient, and paper-free. As our project has come to a close, there is much that we can reflect and improve upon. We struggled a bit with managing our time for this project, but we were able to implement all of the essential functionalities that the I-House management team desired. More features would have been better for our app, but with what we currently have, the I-House management team is satisfied.

Our project manager, Daanyaal, has decided to continue working on the I-House App post-graduation. The continued support will aim to improve and optimize the current functionalities of the web application. The planned social media features and functionalities such as resident messaging will be integrated into the project. There are plans to write support documentation for future teams that are willing to take on ownership of further developing the application. As it stands, we have successfully executed our plans with the I-House Application and are glad to have been given such an educational opportunity.

# References

Charland, Andre, and Brian Leroux. "Mobile application development: web vs. native." *Queue* 9.4 (2011): 20.

Chen, Z. F., Ding, X., Sun, S. P., Shi, L. M., & Sun, Y. X. (2014). Design and Realization of the Student Dormitory Management System Based on RFID. In Applied Mechanics and Materials (Vol. 631, pp. 1447-1450). Trans Tech Publications.

“Design Principles” React. [Online]. Available: https://reactjs.org/docs/design-principles.html.

Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *6*(2007), 2007.

Introducing the MEAN and MERN stacks. (n.d.). Retrieved from <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack>

Morgan, A. (n.d.). The Modern Application Stack - Part 5: Using ReactJS, ES6 & JSX to Build a UI (the rise of MERN). Retrieved from <https://www.mongodb.com/blog/post/the-modern-application-stack-part-5-using-reactjs-es6-and-jsx-to-build-a-ui-the-rise-of-mern>

Pal, P. (2016, November 21). 12 Benefits of Using Node.js For Web Applications. Retrieved from https://think360studio.com/12-benefits-of-using-node-js-for-web-application/

Sharma, S., & Coyne, B. (2017). *DevOps For Dummies, 3rd IBM Limited Edition*(3rd ed.). doi:<https://www.ibm.com/ibm/devops/us/en/resources/dummiesbooks/>

Subramanian, Vasan. Pro MERN stack: full stack web app development with Mongo, Express, React, and Node. Apress, 2017.

Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), 80-83.

Turner, M., Budgen, D., & Brereton, P. (2003). Turning software into a service. *Computer.*, *36*(10), 38-44.

Using a Three-Tier Architecture Model. (n.d.). Retrieved from https://docs.microsoft.com/en-us/windows/desktop/cossdk/using-a-three-tier-architecture-model

Zhang, X., Hu, Y., Lu, Y., & Gu, J. (2011, August). University Dormitory Management System Based on Agile Development Architecture. In Management and Service Science (MASS), 2011 International Conference on (pp. 1-4). IEEE.

# Appendices (Optional)

## Appendix A – Standard Hardware

1. Processor - dual core @ 2.4 GHz (i5 or i7 Intel processor or equivalent AMD) RAM - 8 GB. Hard Drive - 320 GB 5400 RPM hard drive.

## Appendix B – Appendix Title

[Typical example: you can include a specific interface detail here.]