Architecture and Detailed Design Specification

SE-D2-2019- Placement Coordination System

Team Members

Sl.No	Name	USN#
1.	Ninaad R Rao	01FB16ECS232
2.	Midhush Manohar T.K.	01FB16ECS208
3.	Nishant Ravi Shankar	01FB16ECS236
4.	Meghana Ganesh	01FB16ECS205
5.	M Sumukha	01FB16ECS185
6.	Nesara B R	01FB16ECS226
7.	Mohammed Faizan	01FB16ECS211
8.	M Pradeep Kumar	01FB16ECS183
9.	Manish Chandrashekar	01FB16ECS191
10.	Neha B Garg	01FB16ECS224
11.	Omkaram Dhanush	01FB16ECS240

Revision History

Date	Description	Author	Approver	Comments
<date>	<Version 1>	<Your Name>	<Approver's Name>	<First Revision>

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Your Name>	Lead Software Eng.	

Contents

Revision History 1

Document Approval 2

1.0 Architectural and component-level design 4

2.0 System Structure 5

2.1 Architecture diagram 5

2.2 Description for Component 6

2.3 Interaction Diagrams 6

2.4 Describe usage scenarios and how you would test that 7

2.5 Architectural Styles and Patterns considered and for what reason 8

3.0 User interface design 10

4.0 Detailed Design Approach 11

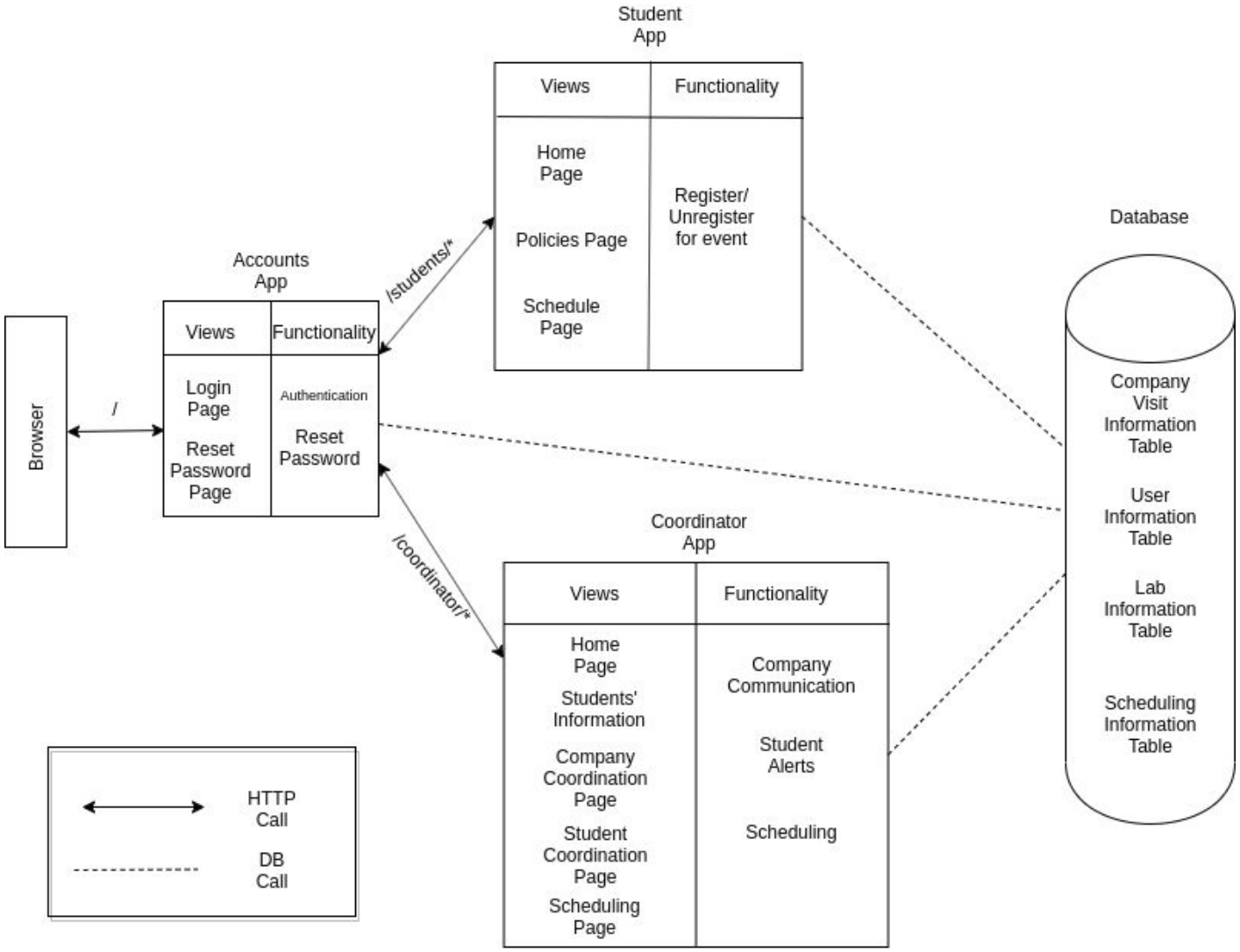
4.1 Design patterns considered and for what reason 11

1.0 Architectural and component-level design

The main actors in the application are the students taking part in the placement process and the placement coordinator. The entire application is divided into 4 components, the Student Component, the Coordinator Component, the Accounts Component, and the Database Component. The division has been made based on the functionalities. The application, as seen by the placement coordinator, is completely different from that seen by the student, along with the different levels of access. Thus, the functionalities executed by the coordinator are different from that of the student, which motivates the division of the application into a Student Component and the Coordinator Component. Except for the Database Component, the components are accessible via rest calls, and the components also communicate with each other via rest calls.

2.0 System Structure

2.1 Architecture diagram



2.2 Description for Component

2.2.1 Browser- It is an application used to access and view the website.

2.2.2 Accounts Application - Contains the login page and password reset page to authenticate users and handle any exceptions by allowing the password to be reset. There are two views for this. One for the admin(placement coordinator) and the other for students. Based on the credentials, the app will redirect to the respective view(student app or coordinator app).

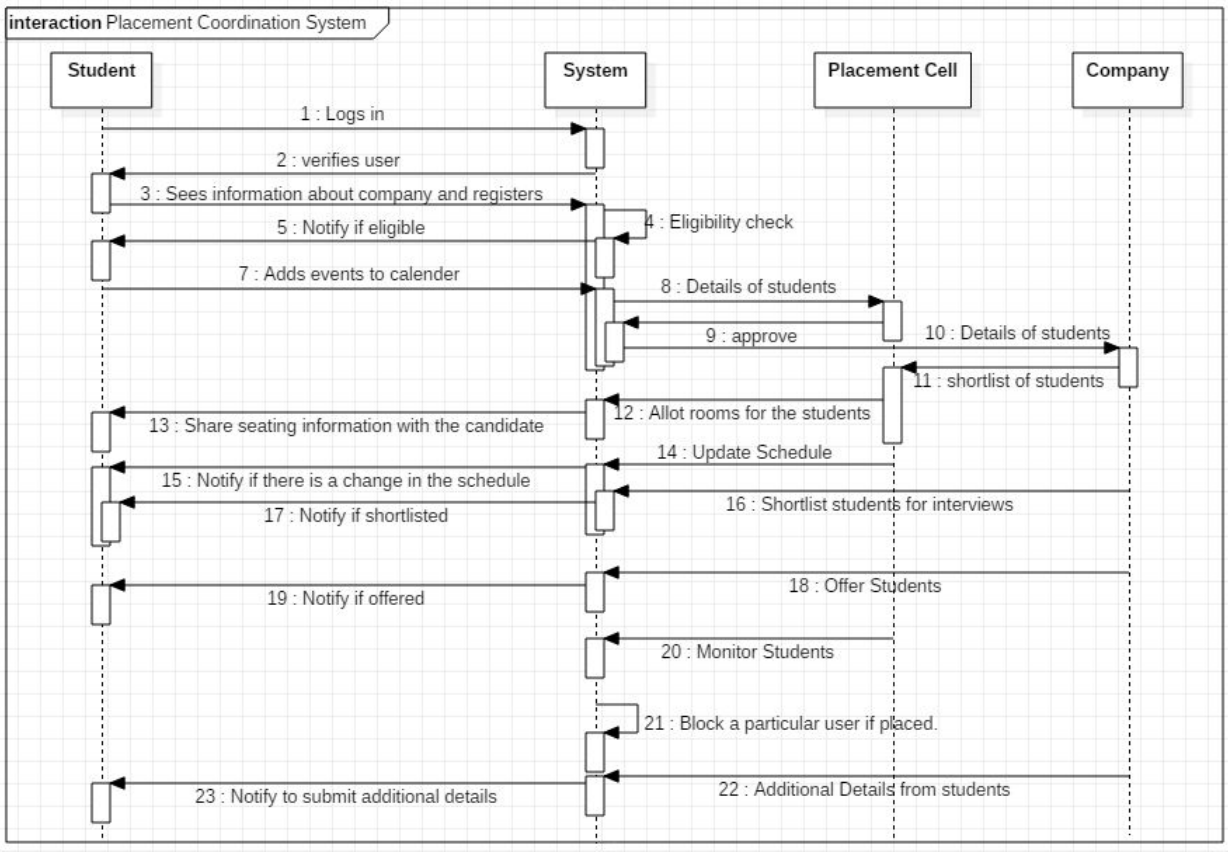
2.2.3 Student Application - The students application contains the student dashboard which displays student information, displays upcoming events, displays the schedule for a particular event. It also includes features like one click registration for students to register for a particular company. There is a feature for seeing placement statistics including the companies that have already come and any other general information that is relevant for a student. It will also have a calendar system to keep track of the events

2.2.4 Coordination Web Application- The coordination web application is for the placement coordinator where he/she can track the students who are part of the placement cycle. There is also a feature for the coordinator to schedule an event by querying the database for finding the desired timeslot. The coordinator can make a particular company details available for the students. The coordinator can share the student details with the company.

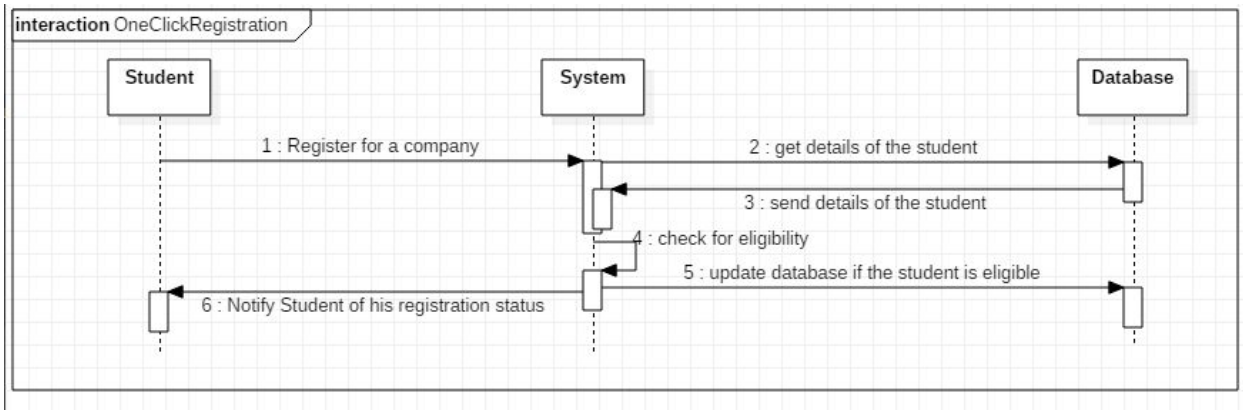
2.2.5 Database-Contains all information regarding students and users, companies,labs and seating, scheduling etc. All the other components will query this particular component to display and update data.

2.3 Interaction Diagram

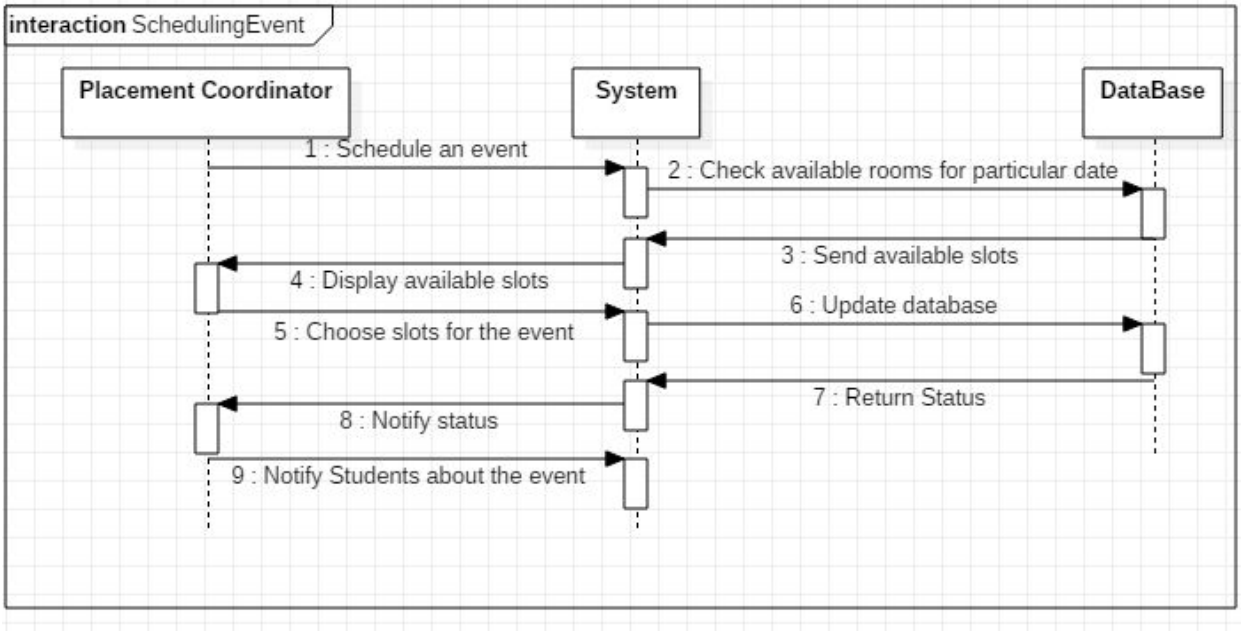
General Flow:



One Click Registration:



Scheduling event:



2.4 Describe usage scenarios and how you would test that

Usage Scenario(STUDENT):

- **REGISTRATION FOR PLACEMENT CYCLE** : User performs a one-click registration for placements, and this action should be recorded in the database.

  A test could be run on the status check of the student on registering for placements, to see if the database action has been performed. This is followed by the uploading of a resume in a specified format. A test to check if the right format of the file has been uploaded will be done here.

- **ONE-CLICK REGISTRATION** : If user wishes to register, he/she can do so just by confirming the registration. Users details will be extracted from the database for further process. If user does not have an offer, registration will be successful or else registration will be unsuccessful.

  All the details will be obtained from the database and appropriate status codes will be checked for both success and failure.

- **NOTIFICATION** :
  - Placement officer will communicate with a company and alert students with the latest updates about the company.
  - Time, venue and seating allotment for online assessment will be sent to the user.
  - Shortlisted for the interview process will be posted.
  - Shortlisted user will be notified about the offer.

Usage Scenario(PLACEMENT OFFICER):

- **SCHEDULING** : schedule the written tests/interviews with respect to time and location.
  A test could be done to verify if the particular time slot is available in the respective computer labs.
- **POST NOTIFICATIONS** :
  - alert students with the latest updates about company.
  - post about online assessment details.
  - post about the results of online process and interview process.
  - Using Postman, we will check the required status and response for a given api calls.

2.5 Architectural Styles and Patterns considered and for what reason

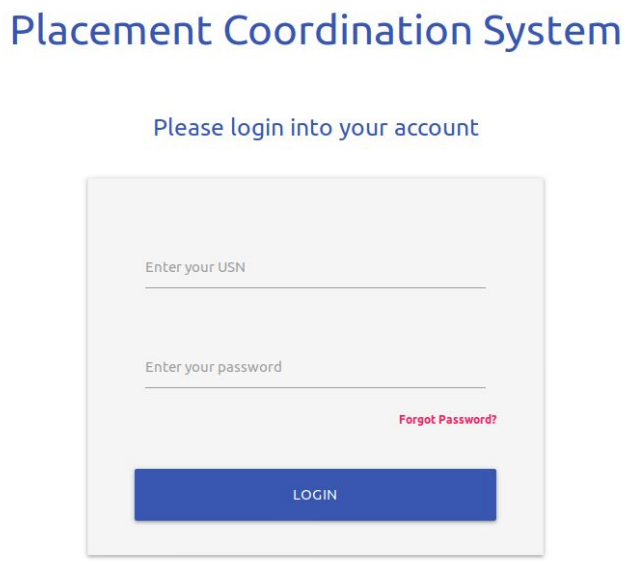
S.No	Architectural Styles/ Pattern	Intent of this pattern	Rationale for choosing or not choosing
1.	Repository Style	A repository architecture is a system that will allow several interfacing components to share the same data. Each component interfaces the same dataset that is utilized system wide. Data manipulation taking	The main intention for using repository style is because our application requires data to be generated by both students as well as placement cell. Students generate data by registering for companies and giving other details. The placement cell generates data by giving information about a particular event(company details and the schedule). These information are exchanged very frequently and thus communicate with the common database.

		place in one component will reflect an identical representation of data in another component. Components can be interchanged and are independent of other system components.	
2.	Microservice Architecture	Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely coupled and independently deployable.	Since the entire system will be used by students and placement coordinators of a college(right now one college), to ensure the system is highly scalable, available, a microservice architecture is used. The main aim is to make sure that a particular service does not choke the other services that are requested by other users(For example, if a particular seating information has been made available and many are accessing it, then it shouldn't avoid the placement coordinator to access or upload some company information.
3.	Pipes and filters	The Pipe and Filter is an architectural design pattern that allows for stream/asynchronous processing. In this pattern, there are many components, which are referred to as filters, and connectors between the filters that are called pipes. Each filter is responsible for applying a function to the given data; this is known as filtering. Filters can work asynchronously. The final output is given to	Pipes and filters architecture is mainly used for large processes that can be broken down into multiple steps. If there needs to be more interaction between functions (the filters), this would not be a good architecture design since filters only communicate between each other for input/output. Since our system will have a lot of components that will be communicating with each other in terms of reading the database, displaying the content and updating the database, we will **not** be considering this architectural style.

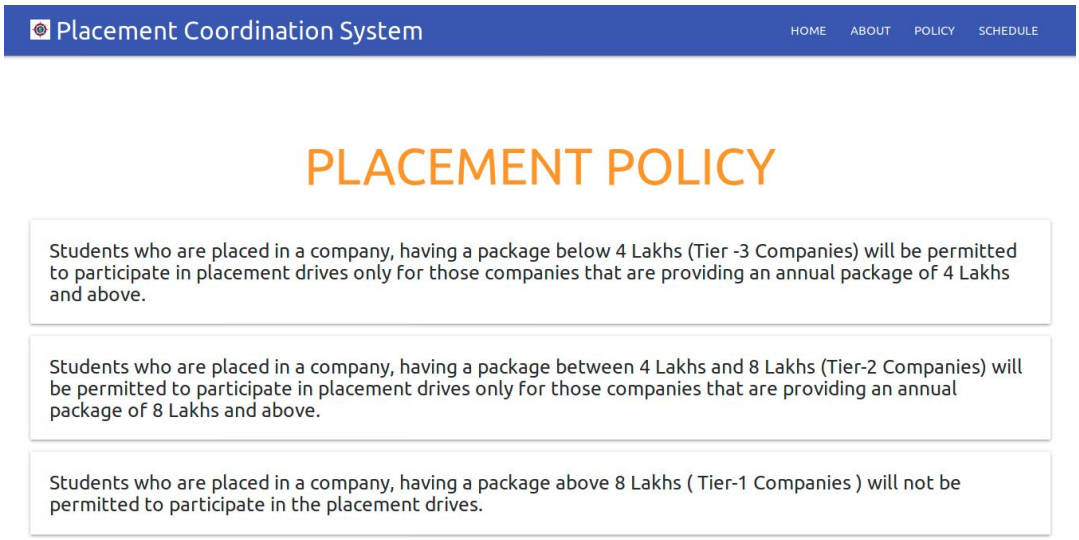
		the consumer, known as a sink.	

3.0 User interface design

A couple of UI elements have been added in the document. These include:



A Login Page - this takes in the USN and password from the user (student/coordinator) and renders the respective homepage.



A Home Page - this is a generic homepage whose layout will be common to both the student as well as the coordinator. The coordinator will have the added functionality to change various elements, including the scheduling details.

4.0 Detailed Design Approach

The project has 2 types of interfaces: Students and Coordinator.

There are backends for each interface, which interacts with the front end. This interaction is via a Model-View-Controller approach, where the front-end is the “View” (what the user sees), the back-end is the “Controller” (what orchestrates the work), and the Mongo database is the “Model” (what stores the information). Whenever a request comes in, it first goes to the Controller before it can be converted into instructions for the View or Model.

The design enabling techniques in the architecture and design of the project is centred around two primary concepts:

- Modularity: The enormity of the project dictates the imperative need for Modularity, which is the degree to which individual components may be separated and recombined. This will aid in flexibility as well as increase independence during the development stage.
- Information Hiding: In general, an abstraction of the interface from the business logic of the internal components is necessary to ensure consistency. The development of this project requires constant modifications to the backend logic, mainly for expansion to incorporate more students and labs, as well as bug fixes and added functionalities. Ensuring appropriate information hiding is therefore a mandatory part of designing components for this project.

4.1 Design patterns considered and for what reason

S.No	Design Pattern	Intent of this pattern	Rationale for choosing or not choosing
1	Structural Design Pattern - Bridge	To separate an object’s interface from its implementation	● Decouple an abstraction from its implementation so that the two can vary independently. ● Different hierarchies for the interface and implementation.
2	Private Class Data	To restrict accessor/mutator access	Students cannot access certain classes of the Coordinator
3	Creational Design Pattern - Object Pool	To avoid expensive acquisition and release of resources by recycling objects that are no longer in use	This design pattern is necessary as the Placement Coordination System’s student base changes every year, which necessitates recycling of resources.

4	Creational Design Pattern - Prototype	To create a fully initialized instance to be copied or cloned	A Prototype of a student is created, which contains properties that are similar for each and every user.
5	Behavioral Design Pattern - State	To alter an object's behavior when its state changes	Once a student gets placed, his/her state needs to reflect that based on several conditions. This will determine his further actions.
6	Behavioral Design Pattern - Strategy	To encapsulate an algorithm inside a class	The scheduling algorithm is built inside the coordinator class for the allotment of lab seats.

5.0 Requirement Traceability Matrix

Sl. No	Req Id	Brief Desc	Architecture Ref Section	Design Ref	Code File Ref	Unit Test Cases	Function/ System test cases
1		Student Information	FR1	2.2.3			
2		Seating allotment for shortlisted students	FR2	2.2.3,2.2.4			
3		Placement Statistics	FR3	2.2.3			
4		Placement Policies	FR4	2.2.3			
5		Maintaining a calendar system for students	FR5	2.2.3			
6		Company information and job search filter	FR6	2.2.3			
7		Tech Events	FR7	2.2.3			
8		One click registration	FR8	2.2.3			
9		Scheduling Event	FR9	2.2.3,2.2.4			