Team Project

Design Report

"Student Assist app"

Course ID: CS487 Software Engineering I

Professor: Dennis Hood

Team Name: Team G8

Team Members:

- Shivdeep Bisurkar (A20525712)
- Prajakta Kumbhar(A20523710)
- Colin Brennan (A20418083)
- Hao-Chih Weng (A20480011)

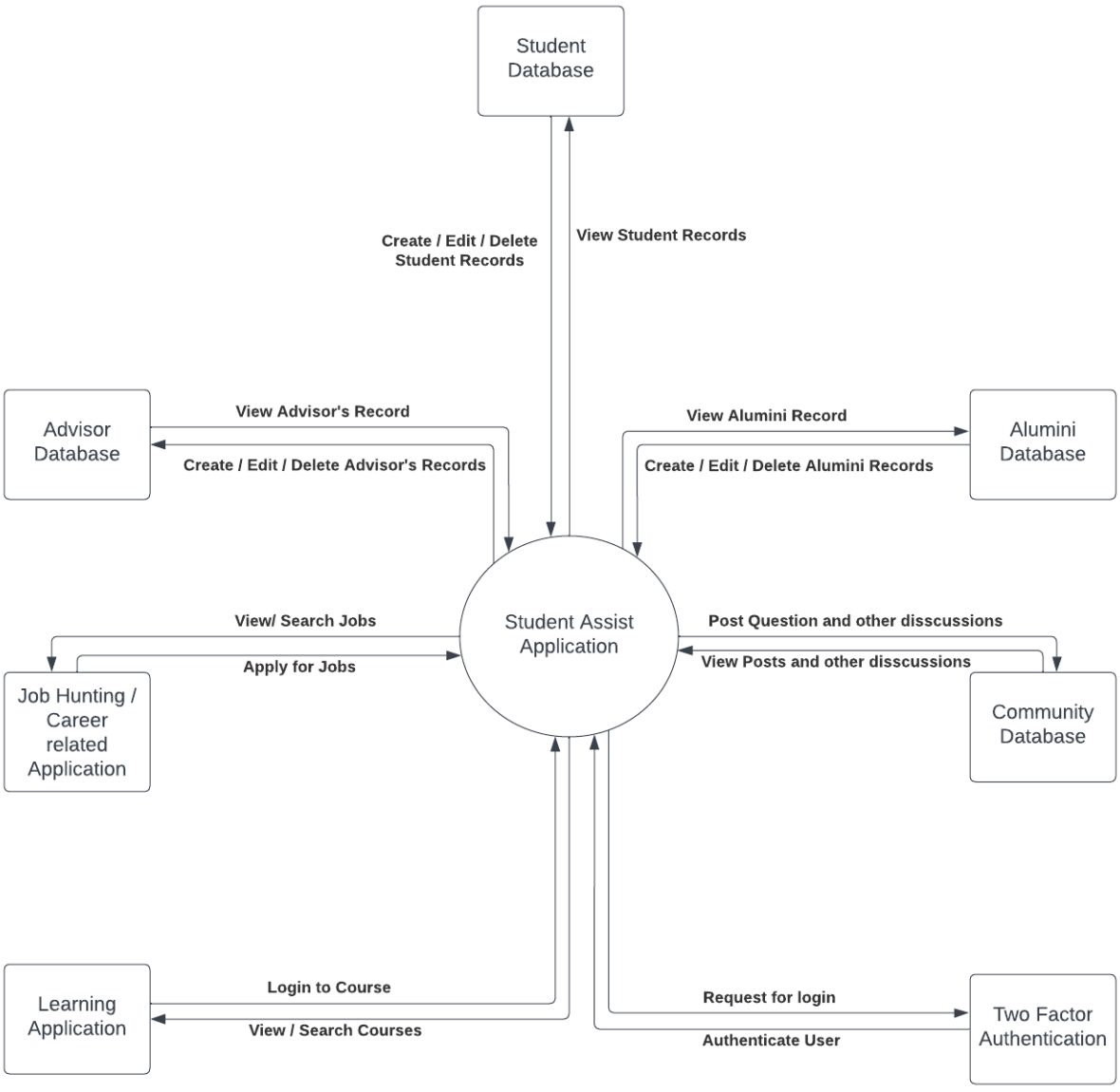
Date: March 25, 2023

Section 1 - Proposed Application Summary

Aside from academic achievements, college students nowadays place a greater emphasis on the role of interpersonal connections in their career prospects after graduation. Developing mobile apps can be a powerful way to enhance both factors, offering opportunities for job matching and networking.

Our team is designing a web-based application to assist students in college and connect students with each other, alumni, and companies for future jobs. Once students have connected with each other they will also have the ability to collaborate and join various communities to have a multitude of discussions. Students also have the ability to contact their advisors or various tutors to set up appointments when necessary. In order to help keep our application secure we plan on implementing the two-factor university login with Google OAuth and Okata. From the homepage, users are able to select a variety of tabs such as their profile, reports/tasks, advisor contacts, assistance, view their connections, collaborate in communities, search for jobs, as well as view a roadmap to their career. Each tab will have different information, tools, and options to assist the students in their daily lives. Overall, the application aims to connect students with each other, alumni, and companies for future job opportunities, as well as provide a platform for students to collaborate, seek assistance, and engage in discussions

Section 2 - Context Model



Student Database: Students can create/edit their account. Admin should be able to Create/ Edit/ Delete students records. Advisor, Alumini should be able to see student records

Alumni Database: Alumni can create/edit their account. Admin should be able to Create/ Edit/ Delete alumni records. Students should be able to connect with Alumni.

Advisor Database : Advisor should be able to Create/ Edit/ Delete alumni records. Students should be able to see the Advisor's record.

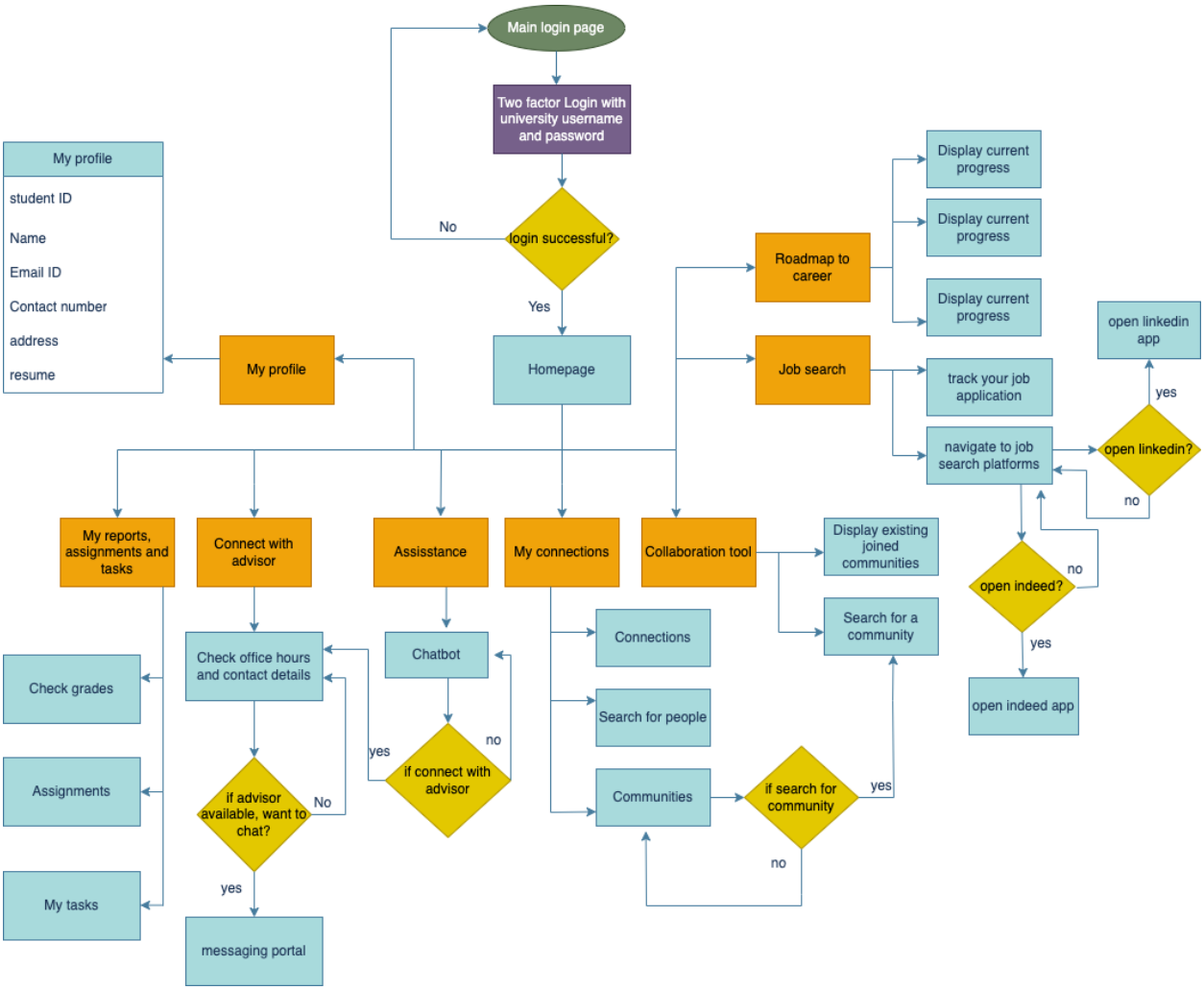
Job Hunting/ Career related application: Students should be able to connect job hunting applications and be able to view company's post and should be able to apply.

Learning Application: Students should be able to see and register for courses on learning applications through student assist applications.

Community Database:students should be able to join various communities like the 'Computer science community', 'sports community', etc where students with the same interests can find, join, collaborate and communicate in that particular community. Options we will provide in this tool: CS community, Law school community, Sports community, SWE (School/Assignment Section, On Campus Event Section, and General Chat Section)

Two Factor Authentication: While login use option like security question, additional code (through mail or text)

Section 3 - UI and User Navigation



Workflow diagram for student assist app

Section 4 - Pseudocode Perspective

1. Login functionality

```
public String studentLogin(String username, String password) {
    // Check if username or password is empty
    if (username.isEmpty() || password.isEmpty()) {
        return "Please enter both username and password";
    }

    // Query the database to retrieve student data based on the username
    Student student = getStudentByUsername(username);

    // Check if the student exists
    if (student == null) {
        return "Invalid username";
    }

    // Check if the password matches
    if (!student.getPassword().equals(password)) {
        return "Incorrect password";
    }

    // Successful login, redirect to main page
    redirectToMainPage();
    return "Successful login, redirecting to main page";
}
```

2. Make an Appointment functionality

```
public void bookAdvisorMeeting(Student student, AdvisorType advisorType, Date meetingDate) {
    // Check if the student is eligible to book an advisor meeting
    if (!student.isEligibleForAdvisorMeeting()) {
        System.out.println("You are not eligible to book an advisor meeting at this time.");
        return;
    }

    // Check if the advisor is available on the requested meeting date
    Advisor advisor = getAvailableAdvisor(advisorType, meetingDate);
    if (advisor == null) {
        System.out.println("The advisor is not available on the requested date. Please choose another
date.");
        return;
    }

    // Book the advisor meeting
    AdvisorMeeting meeting = new AdvisorMeeting(student, advisor, meetingDate);
    advisor.addMeeting(meeting);
    student.addMeeting(meeting);

    System.out.println("Your meeting with " + advisor.getName() + " on " + meetingDate.toString() + "
has been booked.");
```

```
    // Redirect to the message portal
    showMessagePortal(student);
}
```

3. Search community functionality

```
public void enterMyConnections() {
    displayMyConnectionsPage();
    int option = getUserOption();

    switch (option) {
        case 1:
            searchPeople();
            break;
        case 2:
            watchCommunityDiscussions();
            break;
        case 3:
            searchCommunities();
            break;
        case 4:
            viewLinkedPeople();
            break;
        default:
            System.out.println("Invalid option selected.");
            break;
    }
}

public void displayMyConnectionsPage() {
    // Display the My Connections page
    System.out.println("Welcome to My Connections!");
    System.out.println("1. Search People");
    System.out.println("2. Watch Community Discussions");
    System.out.println("3. Search Communities");
    System.out.println("4. View Linked People");
}

public int getUserOption() {
    // Prompt the user to select an option and return their input
    Scanner scanner = new Scanner(System.in);
    System.out.print("Select an option: ");
    return scanner.nextInt();
}

public void searchPeople() {
    // Search for people who have joined the app, as well as students and teachers at the school
    System.out.println("Searching for people...");
}

public void watchCommunityDiscussions() {
    // Display the school's community discussion board
    System.out.println("Watching community discussions...");
```

```
}

public void searchCommunities() {
    // Search for communities
    System.out.println("Searching for communities...");
}

public void viewLinkedPeople() {
    // Display the list of linked people
    System.out.println("Viewing linked people...");
    int option = getUserOption();

    switch (option) {
        case 1:
            cancelLink();
            break;
        case 2:
            blockPerson();
            break;
        default:
            System.out.println("Invalid option selected.");
            break;
    }
}

public void cancelLink() {
    // Cancel the link with a person
    System.out.println("Cancelling link...");
}

public void blockPerson() {
    // Block a person
    System.out.println("Blocking person...");
}
```

4. Link to job-hunting community sites

```
public void openJobSearch(Platform platform) {
    if (platform == Platform.INDEED) {
        openIndeedJobSearch();
    } else if (platform == Platform.LINKEDIN) {
        openLinkedInJobSearch();
        displayAlumniCount();
    } else {
        System.out.println("Invalid platform selected.");
    }
}

public void openIndeedJobSearch() {
    // Display the Indeed job search screen
    System.out.println("Opening Indeed job search screen...");
}
```

```
public void openLinkedInJobSearch() {
    // Display the LinkedIn job search screen
    System.out.println("Opening LinkedIn job search screen...");
}

public void displayAlumniCount() {
    // Retrieve the number of alumni from the company's LinkedIn page
    int alumniCount = getAlumniCount();

    // Display the number of alumni
    System.out.println("The company has " + alumniCount + " alumni on LinkedIn.");
}

public int getAlumniCount() {
    // Query LinkedIn to retrieve the number of alumni from the company's page
    return retrieveAlumniCount();
}
```

Section 5 - Data Model

Data Tables:

Student Table: studentID (int 9), username (varchar 50), password (varchar 50), name (varchar 50), email address (varchar 50), contactNumber(varchar 50), academicStatus(varchar 50), resume (Blob).
Example data: (123456789, "jdoe123", "password123", "John Doe", "jdoe123@email.com", “1234567789", "Alumni")

Advisor Table: advisorID (num), name (varchar 50), email address (varchar 50), areaOfStudy (varchar 50), available Hours (varchar 50).
Example data: (1, "Dr. Smith", "smith@university.edu", "Computer Science", "Monday-Friday 1pm-3pm")

Advisor Meeting Table: meetingID (num), apptTime (Date), studentID (Foreign to Student Table), advisorID (Foreign to Advisor Table)
Example data: (1, "2023-03-24 10:00:00", 1, 1)

Community Table: communityID (num), communityName (varchar 100), description (varchar 200), creationDate (Date)
Example data: (1, "Chess Club", "A community for chess enthusiasts", "2022-01-01")

Community Posts Table: postID (num), communityID (Foreign to Community Table), studentID (Foreign to Student Table), post (varchar 500), creationDate (Date)
Example data: (1, 1, 1,”example post”, "2022-01-01")

Community Members Table: comMemID (num), studentID (Foreign to Student Table), communityID (Foreign to Community Table), joinDate (Date)
Example data: (1, 1, 1, "2022-01-05")

Linked People Table: linkID (num), p1ID (Foreign to student table student id), p2ID (Foreign to student table student id), link_type (varchar 100)
Example data: (1, 1, 2, "Alumni Connection")

Relationships:

Student table:
One-to-many relationship with AdvisorMeeting table through studentID foreign key.
Many-to-many relationship with Community table through CommunityMember table as an intermediary table.
Many-to-many relationship with LinkedPeople table through p1ID and p2ID foreign keys.

One-to-many relationship with Community Posts table through communityID foreign key.

Advisor table:
One-to-many relationship with AdvisorMeeting table through advisorID foreign key.

AdvisorMeeting table:
Many-to-one relationship with Student table through studentID foreign key.
Many-to-one relationship with Advisor table through advisorID foreign key.

Community table:
One-to-many relationship with CommunityMember table through communityID foreign key.
One-to-many relationship with Community Posts table through studentID foreign key.

Community Posts table:
Many-to-one relationship with Community table through communityID foreign key.
Many-to-one relationship with Student table through studentID foreign key.

CommunityMember table:
Many-to-one relationship with Community table through communityID foreign key.
Many-to-one relationship with Student table through studentID foreign key.

LinkedPeople table:
Many-to-one relationship with Student table through p1ID foreign key.
Many-to-one relationship with Student table through p2ID foreign key.

Section 6 - Design Approach

1. **The application can be opened and used from any platform.**
   The application will be designed using cross-platform technologies, such as React Native to ensure that it can be opened and used from any platform.

2. **The app should be loaded within 1 sec.**
   To ensure the app loads within 1 second, the app's architecture should be designed to optimize loading times. This will be achieved through techniques such as code-splitting, pre-loading, and caching.

3. **It should be available 24/7.(no downtime of the application)**
   To guarantee that the application is accessible around-the-clock, it will be hosted on a high-availability server with built-in redundancy. The program can be made available at all times by implementing an automatic failover system and using a load balancer to spread traffic among several servers.

4. **Students should be able to make their profile private and have the ability to share profile with any specific group of users.**
   To ensure that users can make their profile private and share it with specific groups of users, the application will have role-based access control functionality. This will be achieved by assigning roles to users and creating access control lists that specify which roles can access which parts of the application.

5. **The application should be secure. User credentials should be stored in encrypted form.**
   - Use strong encryption algorithms to encrypt user credentials. The encryption algorithm should be difficult to crack, such as AES or SHA-256.

- Use a secure storage mechanism to store user credentials. This could be a database, a file, or a cloud storage solution. The storage mechanism should be secure and properly configured.
- Implement a secure login system that requires users to provide their credentials before accessing the application. This should include a strong password policy and two-factor authentication.
- Implement proper input validation and sanitization, using prepared statements or stored procedures for database queries, and implementing security headers.
- Perform penetration testing, vulnerability scanning, and code reviews.

6. **The application should be scaled properly according to device (i.e. application view on each device should be clear)**

   Use a responsive design approach, test the application on different devices, use appropriate fonts and font sizes, optimize images, use consistent design elements, and use a device detection library

7. **The app should be user-friendly and easy to navigate.**

   Conduct user research, design a clear and intuitive user interface, use a simple and straightforward navigation menu, implement search functionality, provide clear and concise instructions, and conduct usability testing.

8. **Users should be able to seamlessly transition between different pages and functionalities within the application**

   Design a consistent user interface, using a common navigation menu, implementing breadcrumb trails, providing clear feedback, optimizing page loading times, and thoroughly testing the application can be taken.