# **CS3354 Software Engineering Final Project Deliverable 2**

**Airline Crew Scheduler**

Dylan Kaneshiro

Vaidehi Patil

Hiranmayi Tangella

Hruday Rudraraju

Chandini Muthukumar

Fiona Burleson

Austin Polk

**1. [5 POINTS]** Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that in the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student fairly.

| Team Member | Contribution |
| --- | --- |
| Dylan Kaneshiro | * Project Proposal * Project Deliverable 1, Questions 1.5 and 9 * Project Deliverable 2, Question 3 (All parts) * Final Presentation Slides |
| Vaidehi Patil | * Project Proposal * Project Deliverable 1, Question 8 * Project Deliverable 2, Question 6 * Final Presentation Slides |
| Hiranmayi Tangella | * Project Proposal * Project Deliverable 1, Questions 1.3 and 5 * Project Deliverable 2, Question 5 * Final Presentation Slides |
| Hruday Rudraraju | * Project Proposal * Project Deliverable 1, Questions 1.2 and 5 * Project Deliverable 2, Question 4 * Final Presentation Slides |
| Chandini Muthukumar | * Project Proposal * Project Deliverable 1, Question 7 * Project Deliverable 2, Question 8 * Final Presentation Slides |
| Fiona Burleson | * Project Proposal * Project Deliverable 1, Questions 1.4 and 6 * Project Deliverable 2, Question 8 * Final Presentation Slides |
| Austin Polk | * Project Proposal * Project Deliverable 1, Questions 1 and 4 * Project Deliverable 2, Questions 1 and 2 * Final Presentation Slides |

**2.** **[10 POINTS]** Everything required and already submitted in Final Project Deliverable

1. Please specify this part as “Project Deliverable 1 content”.

**Project Deliverable 1 Content:**

1. (**Austin**) [5 POINTS] Please attach here the Final Project draft description (that contains the instructor feedback). It is ok to include a picture of the original document. Address the feedback provided for your proposal by listing what you did / plan to do to comply with those proposed changes and or requests for additions to your project.

Final Project Draft Description:

**Title:** Airline Crew Scheduler

**Group Members:** Dylan Kaneshiro, Vaidehi Patil, Hiranmayi Tangella, Hruday Rudraraju, Chandini Muthukumar, Fiona Burleson, Austin Polk

**Project Description:**

Airline planning is the process of scheduling and allocating resources (including planes, fuel, and crew) for proposed flights, and airline recovery is the process of adjusting airline plans due to delays or other unforeseen circumstances. Airline planning and recovery can be broken down into the sub-tasks displayed in the figure below.

Our app aims to accomplish crew scheduling and crew recovery. After flights are scheduled, our app will connect to the flights database and assign available crew members (pilots, flight attendants) to each flight. Our app will also cleverly adjust schedules in the case of delays or cancellations. Crew members will be able to log into our app, check their schedule, and report any unavailabilities.

**Motivation and Use Cases:**

We have all traveled in an airplane before, and some of us have even been unfortunate enough to have their flights delayed, rescheduled, or canceled. From our viewpoint, it is just one plane that got delayed, rescheduled, or canceled. However, at an airport, there are multiple of such flights scheduled one after the other with a very short frame of time between them. And one delay sets into motion a multitude of rescheduling of flights to accommodate the lost time and a whole new set of challenges in regard to the crew, passengers, co-existing companies, and gate changes. Because we all at some point have experienced the frustrations of flight rescheduling, we mutually decided to build an Airline Planning software which takes care of adjusting airline plans and recoveries in a timely and efficient manner.

This app will be used by airline companies to manage employee schedules and update flight statuses and schedules when flights are delayed as well as by pilots and crew members to view changes to their schedules. Since available flight crew will be scheduled onto flights whose crew is affected by prior delays, these flights will be taking off as scheduled rather than also be delayed.

**Tasks for Each Member:**

● Dylan will research algorithms to perform crew scheduling and crew recovery, and contribute to functional requirements.

● Vaidehi’s task for this project would be to research and set up a system that takes care of the vacancy and availability of the gate or the terminal at which the delayed/rescheduled flight should land at. Flight delays/reschedulings cause shifting of the gate numbers due to multiple flights coming in at the same time, and this algorithm should help avoid such conflicts.

● Hiranmayi’s task for this project involves research regarding interactions between different companies and requirements needed to implement communication between the companies. Contributed to non-functional requirements

● Fiona’s task for this project involves giving real time changes to pilots’ schedules to handle the issue of delays. Setting up a system to handle flights that become vacant due to delays and filling them in with pilots who are available will be achieved by Fiona.

● Austin’s task for this project involves location matching of flight crews to ensure that crews can be scheduled based on availability.

● Hruday’s task for the project is to design the data structure to help optimize the algorithm to schedule the crew and flights. Contributed to functional requirements.

● Chandini will connect the employee schedules so that pilot schedule changes can affect other pilots’ schedules.

Instructor feedback:

*A lovely topic!! Once complete, it will save a lot of time and effort for scheduling crew in airlines: an intensively dynamic platform. Please consider implementing it fully, if you can. No pressure w.r.t. grade on implementation.*

*In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.*

*Fair delegation of tasks.*

*Please share this feedback with your group members.*

*You are good to go. Have fun with the project and hope everyone enjoys the collaboration.*

Response to Feedback:

**Comparison with similar applications:**

Similar applications are [Sabre AirCentre Crew Management](https://www.sabre.com/page/as-product-dictionary/opr-plt-crew-management/) and [Jeppesen Crew Pairing](https://ww2.jeppesen.com/airline-crew-optimization-solutions/airline-crew-pairing/). Both of these applications manage airline crew planning and recovery with proprietary management algorithms that take into account factors such as regulation, company policy, and labor/lodging costs. They are leaders in the industry for the task of assigning flight crews, and our product would provide a similar service to theirs. However, we believe that our project still has value to add in the space. Whereas solutions offered by Sabre may come in several pieces, we can have a more seamless solution which comes as a single package. And whereas Jeppesen’s solution focuses more on behind-the-scenes planning, we will also integrate real-time user interaction and viewing into our product.

**Fair delegation of tasks:**

The team has divided the responsibilities for this deliverable between all the members such that each member can contribute equally to its completion. We have also met as a team in order to discuss the trajectory of the projects and the components required for its success, and we have assigned tasks for each member to complete.

1. [10 POINTS] Setting up a Github repository. Please use your utdallas email accounts only for each group member.

1.1. Each team member should create a GitHub account if you don’t already have one.

(**Hruday**) 1.2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).

(**Hiran**)1.3. Add all team members, and the TA as collaborators. Our TA will post his GitHub info on EL: TA GitHub id: TA email:

(**Fiona**)1.4. Make the first commit to the repository (i.e., a README file with [team name] as its content).

(**Dylan**) 1.5. Make another commit including a pdf/txt/doc file named “project\_scope”. If you choose a predefined topic (one of the 4 topics described in the “Project Topic Ideas” section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.

1.6. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.

Important Note: • Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities. • Do not include credentials (e.g., UTD ID) in the repository. • Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work!

1. [5 POINTS] Delegation of tasks: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.

Labeled with parentheses by each question number

1. (**Austin**) [5 POINTS] Which software process model is employed in the project and why. (Ch 2)

This project will employ the iterative software process model. This will help the team to effectively manage risk and deal with requirements that may be subject to change as the project grows in scope. It also allows the project to be divided up into smaller portions, each of which can be delivered in successive iterations of the product. Between iterations, client feedback can also be gathered in order to improve upon features already implemented in the design and keep track of new features desired by the client(s). With this software process model, our team could also get our product onto the market sooner by first prototyping a minimum viable product and building it up from there with subsequent iterations. Testing and quality control can also be implemented between releases for each iteration, so that the overall product can become more robust and reliable with time.

1. (**Hiran and Hruday)** [15 POINTS] Software Requirements including

5.a.) [5 POINTS] Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

Functional Requirements:

1. Mobile Application for airline staff to report and issues with their own schedule or with an airplane.

2. Authentication of users when they are logging in.

3. Database capable of storing large amounts of data, with quick read and write times (<.1 seconds) The database will store the available planes and staff at the given airport

4. Algorithm that dynamically reassigns planes and crews based on any issue that occurs such as an issue with a plane or crew.

5. APIs to receive information about issues with planes, and flight crew

6. API’s that update based on flight plans and routes when airplanes arrive at the airport, and when they depart to update databases with available staff and airplanes.

7. API’s alerting flight staff of any updates to there schedule from the standard or pre decided schedule due to unforeseen circumstances.

5.b.) [10 POINTS] Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one non- 4 functional requirement for each of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements to be able to provide one for each. Please explicitly specify if you are considering such assumptions.

**Product requirements**:

1. Usability requirements: Must be easy to use for the crew and staff.

2. Efficiency requirements:

- Performance requirements: The final product must not cause any overlaps in schedule and must readjust immediately on receiving information.

- Space requirements: Must have a large amount of database storage to allow storing of data from multiple different airlines and crew members.

3. Dependability requirements: Should follow the CIA principles (confidentiality, integrity and availability). Outsiders must not know information stored in the software, the information cannot be changed by anyone but a licensed manager from the FAA (assumption), and the software must be available to all crew members with a valid license. The software must also be maintainable to allow for changes in case something goes wrong.

4. Security requirements: Should be able to keep each crews schedule private and out of the view of other people unless another crew member shares the same flight, then it will only show the crew's schedule of that particular flight to the other crew members. Outsiders should not have any knowledge of this as this must be kept confidential.

**Organizational requirements**

1. Environmental requirements: The members who use this software must identify themselves with a given ID or credentials to gain access.

2. Operational requirements: The system will be used only by the crew of each airline and the members who will push changes to schedules in case any changes must be made. It is mostly used to help crew know what flight they will be on, see changes to their flights and also help airlines make sure the number on staff members on board meets the requirements. The crew members can also report on the system if they are unable to work on the flight in case of emergencies.

3. Developmental requirements: Must use different languages such as (C++ or JAVA), JavaScript, HTML, CSS and SQL.

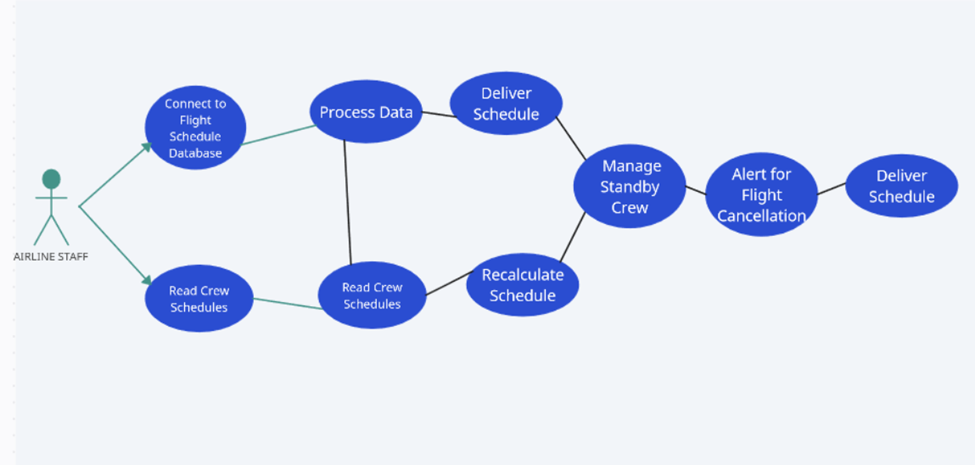
**External requirements:**

1. Regulatory requirements: must have a special ID or authentication to be able to manage or regulate tasks. (assumption) Must not be affiliated with any specific airline and must be part of the FAA (federal aviation administration)

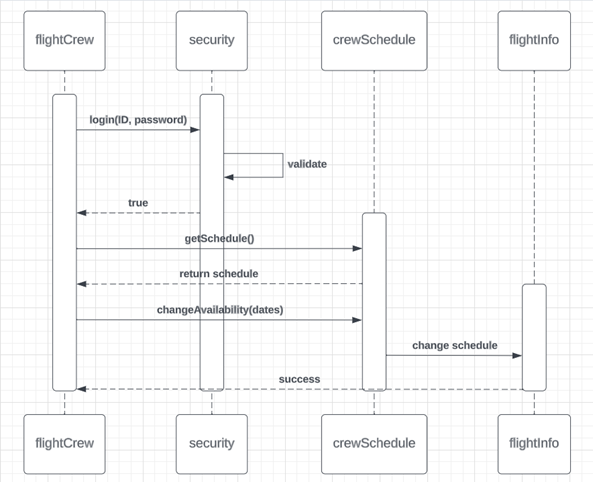
2. Ethical requirements: (assumption) the system administrator must not change schedules to their whim, the crew cannot change or access the schedule of other crew members unless they share a flight, then the details of the flight can be shared. This information cannot be tampered with or shared in any form.

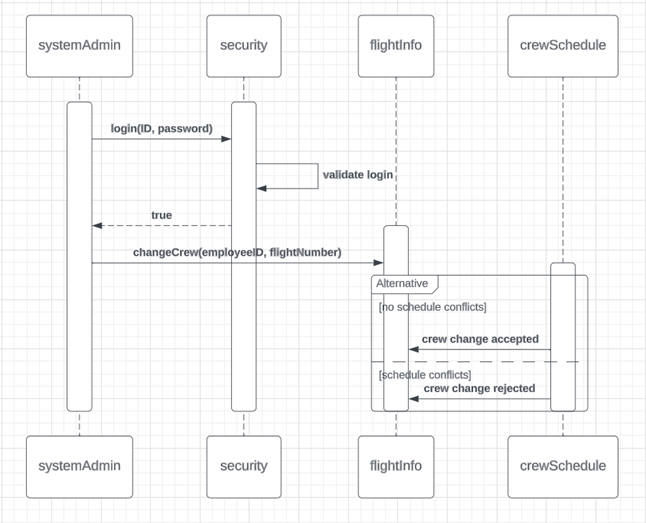
3. Legislative requirements: All members using the scheduling software must have related licenses authorized by the FAA. Also the software must follow rules from 14 CFR 91.533 which includes whether a person may operate an airplane or not based on the ratio of passengers to crew members.

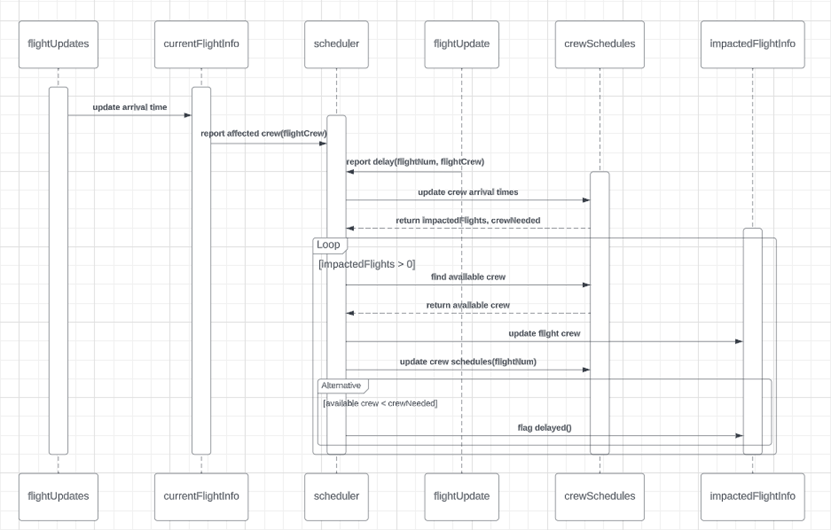
1. (**Fiona**)[15 POINTS] Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note than there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)



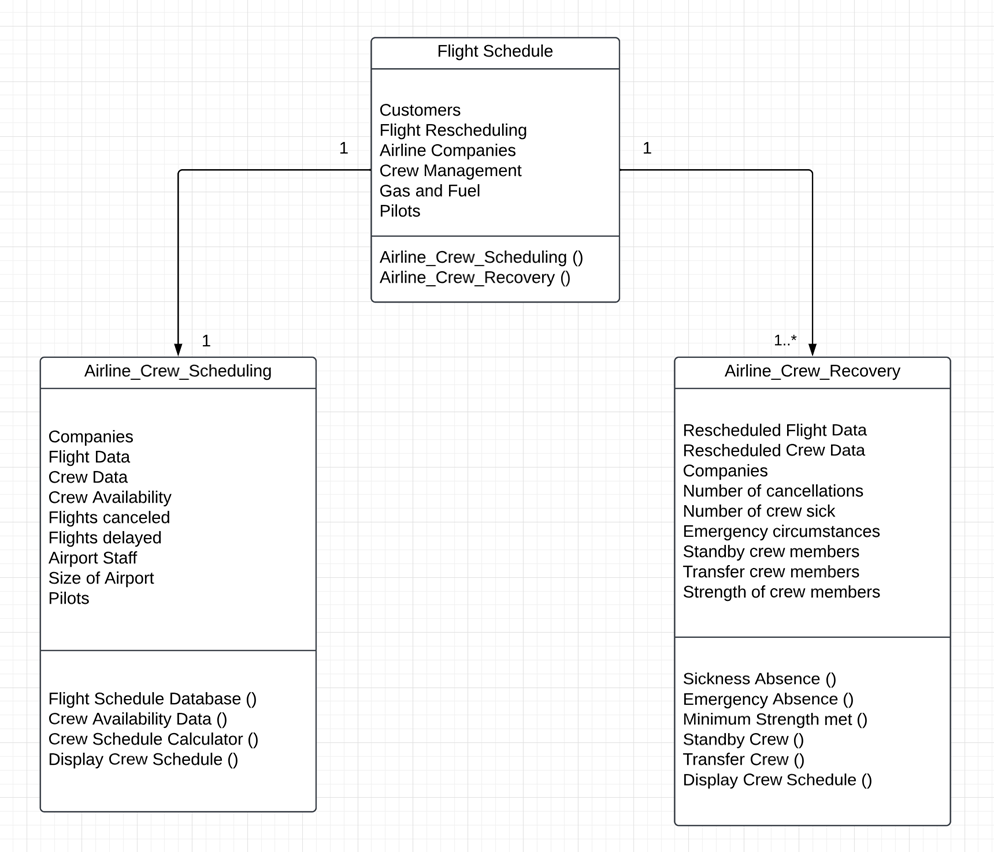
1. (**Chandini**) [15 POINTS] Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)







1. (**Vaidehi**)[15 POINTS] Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



1. (**Dylan**) [15 POINTS] Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3)

9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6)

9.2. Layered architecture pattern (similar to Figure 6.9)

9.3. Repository architecture pattern (similar to Figure 6.11)

9.4. Client-server architecture pattern (similar to Figure 6.13)

9.5. Pipe and filter architecture pattern (similar to Figure 6.15)

Code for Model, View, and Controller classes will be heavily modularized

Our app will not just be downloaded by companies. Ideally, companies would consult with us and we would build an app that is tailored to their specific business needs and constraints, in a way similar to Oracle applications. The foundation of the app is described below in terms of the MVC architecture.

Our **Model** Class will handle the data related to flights and crew schedules

- Data tables

- User login

- Columns = EmployeeID, Name, Password, Job(Pilot, Flight Attendant, or System Admin)

- Flights

- Columns = Flight Number, Airplane Identifier, Departure Airport, Departure Gate, Departure time, Arrival Airport, Arrival Gate, Arrival Time

- This table will be populated by a process that pulls flight data from the specific client’s database. This process will be handled outside of the app, and its implementation will vary for each client.

- Crew Availability

- Columns = Unavailability start, Unavailability end

- Crew Schedule

- Columns = EmployeeID, Flight Number

- Data transformations:

- Schedule Crew for Flight

- Input = Flight Number, Flights database, Crew availability, Crew Schedule

- Output = Modified Crew Schedule

- Schedule Crew for Flights

- Input = Flight Numbers, Flights database, Crew availability, Crew Schedule

- Output = Modified Crew Schedule

- Algorithm = Column Generation

- Paper:<https://www.researchgate.net/profile/Chen-Shijun/publication/237074620_An_Improved_Column_Generation_Algorithm_for_Crew_Scheduling_Problems/links/0deec51b3f06611f3c000000/An-Improved-Column-Generation-Algorithm-for-Crew-Scheduling-Problems.pdf>

- Video:

<https://www.youtube.com/watch?v=YDqegcUu5Y4>

- Cancel Crew for Canceled Flight

- Input = Flight Number, Flights database, Crew availability, Crew Schedule

- Output = Modified Crew Schedule

- Edit Crew for Flight

- Input = Flight Number, Flights database, Crew availability, Crew Schedule, Crew member to remove

- Output = Modified Crew Schedule

- Add / Remove from Crew Schedule and Crew Availability

- Input = EmployeeID, Unavailability Start time, Unavailability End time, Crew Schedule, Crew Availability

- Output = Modified Crew Schedule

- Query functions for all tables

- Combine crew schedule and availability into calendar

- Input = EmployeeID, Crew Schedule, Crew Availability

- Output = Calendar

- Validate Assignment

- Ensures a particular assignment of a crew member to a flight meets constraints

- Meet legal requirements

- Must have particular number of standby crew at each location in case anything goes wrong

- Input = Employee IDs, Flight Number

- Output = True, False

- Validate Login

- Input = Username, Password

- Output = True, False

Our **View** Class will dynamically display html depending on signals it receives from the controller

- All users login page

- If login successful:

- Pilot / Flight Attendant View

- View schedule as calendar

- Button to add unavailabilities

- Button to cancel unavailabilities

- System Admin

- Interface to manually recalculate flights

- Interface to manually add / remove crew members to flights

- Interface to add / remove users

Our **Controller** Class will act as a medium between our model and view

- When user clicks submit login, validate login from data model and pass to view

- When user requests to view schedule, call calendar function of data model and pass to view

- When user requests to insert / delete availabilities, call functions to insert and delete by value in data model and refresh view

- When system admin calls recalculate flights, run schedule flights function and refresh view

- When system admin calls to edit a table, edit table in data model and refresh view

## **IMPORTANT NOTE: The following items will all need to be calculated / worked on based on the project you are designing. As an example, if a team of 7 students in CS3354 class is working on the development of a hospital information system, this group will prepare the project scheduling, cost, effort and pricing estimation calculations based on the hospital information system design, NOT based on their 7 student team. Think of the analogy to the “*Inception*” movie: What you will be working on is the dream in a dream, i.e. the dream in the second level, NOT in the first level.**

**3.** **[35 POINTS]** Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:

3.1. **[5 POINTS]** Project Scheduling. Make an estimation on the schedule of your project. Please provide start date, end date by giving justifications about your estimation. Also provide the details for:

* Start date: January 8, 2024
* Project Duration: 11 weeks (calculated by FP method)
* End date: March 22, 2024
* Whether weekends will be counted in your schedule or not:

No, employees will not work on weekends

* What is the number of working hours per day for the project:

Working hours per day dedicated to the project is 6 out of the 8 working hrs/day

Schedule:

Our development team of 20 will be divided into 4 groups that will handle: Constructing the data model, constructing data transformation algorithms, constructing UI, and constructing the Controller. The schedule will be broken down into 2 week sprints.

| Date | Data Model | Algorithms | UI | Controller |
| --- | --- | --- | --- | --- |
| 1/8 | Create and test secure login table and simple transformations (Create, Read, Update, Delete, while maintaining functional constraints) | Begin researching flight scheduling algorithm (column generation algorithm for one flight + batch of flights) | Create login page | Create login controller |
| 1/22 | Create flights table | Continue | Create pilot / flight attendant view | Create view schedule controller |
| 2/5 | Create and test crew schedule table | Begin implementation | Continue pilot / flight attendant view | Create insert / edit availabilities controller |
| 2/19 | Create and test crew availability table (+ query function to union schedule and availabilities) | Continue implementation | Create system admin view | Create recalculate flights controller |
| 3/11 | Integrate functionality | Finalize Implementation | Integrate functionality | Create system admin controller |
| 3/22 | Project complete | Project complete | Project complete | Project complete |

3.2. **[15 POINTS]** Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Please choose one of the two alternative cost modeling techniques and apply that only:

- Function Point (FP)

|  | Function Category | Count | Complexity | | | Count x Complexity |
| --- | --- | --- | --- | --- | --- | --- |
| Simple | Average | Complex |
| 1 | Number of user input | 6 | 3 | 4 | 6 | 36 |
| 2 | Number of user output | 6 | 4 | 5 | 7 | 30 |
| 3 | Number of user queries | 2 | 3 | 4 | 6 | 8 |
| 4 | Number of data files and relational tables | 4 | 7 | 10 | 15 | 60 |
| 5 | Number of external interfaces | 4 | 5 | 7 | 10 | 40 |
|  | | | | | GFP | 174 |

6 User inputs: crew add unavailability, crew cancel unavailability, manually recalculate flights, manually edit crew members on flight, add users, remove users

6 User outputs: View calendar, view list of unavailabilities, admin view all tables (4)

2 User queries: username, password

4 relational tables: user login, flights, crew availability, crew schedule

4 external views/interfaces: login, pilot/crew, admin, connect to flights database

(From Question 9 of Deliverable 1)

(1) Does the system require reliable backup and recovery? 5

(2) Are data communications required? 5

(3) Are there distributed processing functions? 3

(4) Is performance critical? 4

(5) Will the system run in an existing, heavily utilized operational environment? 5

(6) Does the system require online data entry? 5

(7) Does the online data entry require the input transaction to be built over multiple screens or operations? 0

(8) Are the master files updated online? 3

(9) Are the inputs, outputs, files, or inquiries complex? 1

(10) Is the internal processing complex? 5

(11) Is the code designed to be reusable? 5

(12) Are conversion and installation included in the design? 5

(13) Is the system designed for multiple installations in different organizations? 5

(14) Is the application designed to facilitate change and ease of use by the user? 4

Total PC = 55

PCA = 0.65 + 0.01\*(55) = 1.20

FP = GFP x PCA = 174\*1.20 = 208.8

E = FP / (10 FP per person-week) = 20.88 person-weeks

Note: we are estimating a seemingly low FP/person-week due to the large amount of data transformations that must be handled and the complexity of our project

Since we have a dev team of 20, the FP method tells us that we can complete the project in 20.88 / 20 = 1.044 weeks. However, not all the work is entirely parallelizable, and scaling our workforce by 20 will not result in a 20x productivity boost. Assuming that half the work of our project is parallelizable and the other half is necessarily serial, the amount of weeks in our project becomes 20.88/2 + 20.88/(2\*20) = 10.96 = 11 weeks. This provides a more realistic timeframe for the development of our project.

3.3. **[5 POINTS]** Estimated cost of hardware products (such as servers, etc.)

High-performance laptop: Lenovo ThinkPad X1 Carbon Gen 11 (2023): $1212.99 each

Monitor: Philips 221V8LN : $69.99 each

Keyboard: Logitech MX keys combo for business: $199.99 each

HDMI cable: Amazon basics: $7.68

Adapter: Usb 3.0 Hub Vienon 4-port Usb Hub Usb Splitter Usb Expander for Laptop: $6.35

20 \*(1212.99 + 69.99 + 199.99 + 7.68 + 6.35) = $29,940

We will not be hosting our app since we are a Software as a Service company. Thus, we will not need servers.

3.4. **[5 POINTS]** Estimated cost of software products (such as licensed software, etc.)

IDE (IntelliJ): $599 per user per year

Communication platform (Microsoft 365 Apps for business): $8.25 per user per month

Collaboration platform (JIRA): $8.15 per user per month

Version control (GitHub Team): $3.67 per user per month

Timeline = 2.5 months = 0.2 years

Effective Total Cost = 20\*(599\*0.2 + 8.25\*2.5 + 8.15\*2.5 + 3.67\*2.5) = $3399.50

3.5. **[5 POINTS]** Estimated cost of personnel (number of people to code the end product, training cost after installation)

11 weeks x 40 hrs/week x $45/hr = $19,800

**4.** **[10 POINTS]** A test plan for your software: Describe the test plan for testing minimum one unit of your software. As an evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as additional document in your zip file submitted.

**5.** **[10 POINTS]** Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make.

There is a lot of software out there that is similar to ours when it comes to team scheduling. Some examples are: connectteam, genesis, UKG scheduling software and so on. All these software have one goal in mind: effective planning and scheduling for employees in the field. However most of these scheduling softwares are targeted towards on type of job. For example, connectteam and UKG are more suited for businesses and corporate environments, while Genesis was made for space personnel. The same way, our system works for specifically commercial airline crews. Unlike other professions where schedules change once a week or month, airline crew schedules are constantly changing everyday due to delays, flight changes and emergencies.

There are similar applications in the airline field as well. [Sabre AirCentre Crew Management](https://www.sabre.com/page/as-product-dictionary/opr-plt-crew-management/) and [Jeppesen Crew Pairing](https://ww2.jeppesen.com/airline-crew-optimization-solutions/airline-crew-pairing/) manage airline crew planning and recovery with proprietary management algorithms that take into account factors such as regulation, company policy, and labor/lodging costs. Both these applications are leading in the industry of airline scheduling. They take “costs, crew efficiency, fatigue, operational robustness, crew satisfaction” into account. Sabre has numerous services that are separated ( such as crew access, crew control, recovery crew and so on.) Jeppesen mostly deals with planning behind the scenes, which means the crew doesn't really have an app to see their entire schedule. However, our application aims to combine all the features provided by Sabre into a more concise model that will help pilots and crew.

Our project deals with the schedules of the crew and backup crew. Each member of the crew can view their schedules with the flight details and the crew details for that particular flight. The crew members can also apply for days off, or can take emergency leave. The schedules are changed by an authorized person in 2 circumstances: when the crew size is less than the required amount or when there is a delay in the flight schedule. When the flight is delayed, the crew’s schedule also changes to reflect the changes of the delay. Since a single flight delay may also cause changes to other flights, the changes are reflected in the schedule of all the affected crew members. When the number of crew members is lower than usual, the backup crew is assigned to fill the role of the crew member that is unable to come. In airlines, sometimes the crew of one airline company can be requested to take over the crew of another airline company. Our scheduler can also help with such changes as well.

**6.** **[10 POINTS]** Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

As already described earlier, we are making an Airline Planning software that helps reschedule the Crew and manage the delaying and rescheduling of the flights. Basically, Airline Planning is the process of scheduling and allocating resources (including planes, fuel, and crew) for proposed flights, and Airline recovery is the process wherein the airline plans are adjusted due to unforeseen circumstances such as delays and/or cancellations. Our software’s objectives were to accomplish crew scheduling and crew recovery. The goal was that when a flight gets canceled or rescheduled, our application should be able to connect to the flights database and re-assign the available crew members, included but not limited to the pilots and flight attendants, to each flight. Additionally, our application should also be able to adjust crew schedules cleverly and efficiently in case of flight delays or cancellations. Our Airline Planner software also aimed at offering services where the crew members will be able to log into the system, check their schedules, and report any unavailability.

There were no such major changes in our project building from then to now. Each step in the software building was thoroughly discussed, well thought out, and precisely implemented. There were no alterations in our basic objectives as expected from the software. The only minor change was the reduction of the scale of our project. When the Airline Planning system is mentioned, flight rescheduling due to cancellations or delays, and in turn an altered schedule of airport terminals and gates are also normally considered a part of the project. However, as we began implementing the software for the project, we realized that the scale of the project was too broad and wide, and hence, we needed to narrow it down to Crew Scheduler, which still comes under the umbrella of Airline Planning System. There were thorough discussions and logical opinions were sought to decide on the best Software Architecture Pattern and Algorithmic Estimation Model through previous assignments which allowed us to implement our software project idea concisely and precisely. Therefore, overall, there were no major changes made in the implementation from the very beginning to the end, except for some minor alterations here and there which were simply the building blocks of our project to bring it to completion.

However, in terms of usages, the use cases of our Airline Planning software have changed as we narrowed the scope of our project. Our application now will be used by Airline companies to only manage employee schedules and allow the pilots and crew members to view changes to their schedules and will not update flight statuses and schedules when flights are delayed. Nevertheless, the primary objective of our project is being implemented and has promising usage cases.

**7.** **[5 POINTS]** References: Please include properly cited references in IEEE paper referencing format. Please review the IEEE referencing format document at the URL:

[1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, and P. H. Vance, “Crew scheduling,” SpringerLink, <https://link.springer.com/chapter/10.1007/978-1-4615-5203-1_14> (accessed Nov. 13, 2023).

[2] “Employee scheduling,” UKG, <https://www.ukg.com/solutions/employee-scheduling> (accessed Nov. 13, 2023).

[3] “The World’s #1 employee app,” Connecteam, <https://connecteam.com/> (accessed Nov. 13, 2023).

[https://ieee- dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf](https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf)).

It means that your references should be numbered, and these numbers properly cited in your project report.

**8.** **[10 POINTS]** Presentation slides. No min/max number of slides enforced. Please make sure that you can complete presentation within **20 (twenty)** minutes.

Following template could be a good start to prepare your presentations. As each project topic is different, a variety in presentation style is expected and welcome.

- Title of your project together with participants

- Objective of the project designed

- Cost estimation

- Project timeline (timeline of the project designed, NOT the time you’ve spent on it)

- Functional and non-functional requirements. If too long, select representative items.

- Use case diagram

- Sequence diagram for a selected representative operation of the project.

- Class diagram

- Architectural design

- Model-View-Controller (MVC) pattern (similar to Figure 6.6)

- Layered architecture pattern (similar to Figure 6.9)

- Repository architecture pattern (similar to Figure 6.11)

- Client-server architecture pattern (similar to Figure 6.13)

- Pipe and filter architecture pattern (similar to Figure 6.15)

- Preferably a demo of user interface design that shows screen to screen transitions though no full functionality is required.

- OPTIONAL: IF implemented the project, a demo of your implementation.

## **9.** **OPTIONAL PART [POSSIBLE EXTRA CREDIT UP TO 10 POINTS].** Your

program code **(if fully implemented the project, not required otherwise).** Please note that **implementation is not required for the final project**. Groups are welcome to implement their work, if they choose to do so.

## **[This part may qualify for extra credit, if you implement and submit the implementation code together with your project. The extra credit will be determined based on the quality of your implementation.**

**Furthermore, any fully implemented project qualifies for scholar publication afterwards. This most probably will involve further commitment to work more an write a scholar paper to send to a Conference for publication.]**

**10.** **[5 POINTS]** GitHub requirement:

Make sure at least one member of your group commits everything for project deliverable 2 to your GitHub repository, i.e.

- Your final project deliverable2 report

- Unit test code for a sample unit of your project

- Implementation code (if you have implemented your project)

- Presentation slides

Still, one member of your team should also submit the required project deliverable 2 materials to eLearning.

**Making life easy when working as a group:**

It is very important to make sure that you communicate and share common work with your teammates. Here are some URLs to help you on that:

[Githu](https://github.com/)b — a web-based Git or version control repository and Internet hosting service. This is the recommended version control software for this project. If by some reason you cannot use the Github platform, you may use any of the following similar platforms for sharing your project related material.

·  [Doodle](http://www.doodle.com/)—a tool for time management and meeting scheduling.

·  [GroupMe](http://groupme.com/)—a group messaging service that lets you be in touch with your team members via mobile phones.

·  [CVS, open source version control](http://www.nongnu.org/cvs/) - helps you work on different versions of the same product and merge your versions.

·  [Slack](http://slack.com/) — a web-based team communication service.

· Mercurial [https://www.mercurial-scm.org](https://www.mercurial-scm.org/)/ for version control

# **About Presentation of your Project:**

You may use any style in your presentations. A slide show is recommended as it helps display highlights of your talk to the audience and to yourself. The following is a list of suggested content for your presentations:

A brief introduction to your project topic List of requirements

Use case diagram that contains use cases Sequence diagram – if too many, one of them only Design Class Diagram (DCD)

User Interface Design

Comparison with similar work (if any), or emphasizing its significance and uniqueness (if there is not any)

Conclusion and Future Work

You are welcome to enhance the minimum content listed above, provided that you stay within maximum 20 minutes presentation time requirement. We will listen to multiple presentations per day, so please try not to exceed the allocated time for your presentations so as not to steal from other groups’ presentation time.

Feel free to enrich your presentation with supporting figures, charts, documents, tables, similar work, etc.

Contribute from yourselves: Employ your own design layouts, color selections, animations, artistic perspectives to your presentations. Try to make them attractive. Think of commercials: We only remember the “interesting” ones.

It is a suggested tactic that in a presentation, each slide should remain min. 1 minute on display so that everybody reads and understands it. So, not too many slides maybe a good idea to start with.

Rehearsal will prevent unexpected surprises. Make sure you rehearse and time your presentation before you actually present it.

# **What to submit?**

Please zip the following as one single file:

- Final project deliverable2 report (Please note that your deliverable2 report should also include your deliverable1 report as required in section 2 above.)

- Test code (section 4 above)

- Presentation slides (section 8 above)

- [Optional] If you have fully implemented your project, include your implementation code (section 9 above)