metin, böcek, omurgasız, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

|  |  |  |  |
| --- | --- | --- | --- |
| Approver Name | Title | Signature | Date |
| Cengizhan Aydogan | Requirements Analyst | siyah, karanlık içeren bir resim  Açıklama otomatik olarak oluşturuldu | 12.10.2024 |

TABLE OF CONTENTS

[1.Introductıon 3](#_Toc180356885)

[1.1 Purpose 3](#_Toc180356886)

[1.2 Scope 4](#_Toc180356887)

[1.3 Definitions, Acronyms, and Abbreviations 4](#_Toc180356888)

[1.4 References 4](#_Toc180356889)

[1.5 Overview 4](#_Toc180356890)

[2.General Descrıptıon 4](#_Toc180356891)

[2.1 Product Perspective 4](#_Toc180356892)

[2.2 Product Functions 4](#_Toc180356893)

[2.3 Users and Characteristics 5](#_Toc180356894)

[2.4 General Constraints 5](#_Toc180356895)

[2.4.1 Technical Constraints 5](#_Toc180356896)

[2.4.2 Course Scheduling Constraints 5](#_Toc180356897)

[2.4.3 Instructor Scheduling Constraints 6](#_Toc180356898)

[2.5 Assumptions and Dependencies 6](#_Toc180356899)

[2.6 Apportioning of Requirements 6](#_Toc180356900)

[3.The Specıfıc Requırements 7](#_Toc180356901)

[3.1 User Interface 7](#_Toc180356902)

[3.2 Functional Requirements 7](#_Toc180356903)

[3.2.1 Receiving User Input 10](#_Toc180356904)

[3.2.1.1 Description and Priority 10](#_Toc180356905)

[3.2.1.2 Stimulus/Response Sequences 10](#_Toc180356906)

[3.2.1.3 Functional Requirements for Receiving User Input 10](#_Toc180356907)

[3.2.2 Data Integration 10](#_Toc180356908)

[3.2.2.1 Description and Priority 10](#_Toc180356909)

[3.2.2.2 Stimulus/Response Sequences 10](#_Toc180356910)

[3.2.2.3 Functional Requirements for Data Integration 10](#_Toc180356911)

[3.2.3 Generating Course Schedule 11](#_Toc180356912)

[3.2.3.1 Description and Priority 11](#_Toc180356913)

[3.2.3.2 Stimulus/Response Sequences 11](#_Toc180356914)

[3.2.3.3 Functional Requirements for Generating Course Schedule 11](#_Toc180356915)

[3.2.4 Delivering User Output 11](#_Toc180356916)

[3.2.4.1 Description and Priority 11](#_Toc180356917)

[3.2.4.2 Stimulus/Response Sequences 11](#_Toc180356918)

[3.2.4.3 Functional Requirements for Delivering User Output 11](#_Toc180356919)

[4.Other Nonfunctional Requirements 12](#_Toc180356920)

[4.1 Performance Requirements 12](#_Toc180356921)

[4.2 Software Quality Attributes 12](#_Toc180356922)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| SRS for Bee Plan CSS | 12.10.2024 | First version created. | 1.0 |

# Introduction

This section provides a general overview of the Bee Plan Course Schedule System software project and its objectives.

## Purpose

The purpose of this SRS document is to define the requirements of the Bee Plan Course Schedule System software. It provides information about the target audience and the intended functionality of the software.

## Scope

The Çankaya University Course Scheduling System is designed to streamline the scheduling process for courses across all departments, ensuring that programs are efficiently planned for undergraduate classes from the 1st to the 4th year. The system integrates shared scheduling criteria provided by the Student Affairs Office and takes into account faculty availability, external schedules, and other relevant factors.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Term | Definition |
| Bee Plan CSS | Abbreviation of Bee Plan Course Schedule System. |

## References

1. "IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no.,pp.1-26, 10 Feb. 1984, doi: 10.1109/IEEESTD.1984.119205.

## Overview

This SRS contains all requirements and constraints of Project Bee Plan CSS.

* Section 1 provides an overview of the project and its purpose, defining its scope, objectives, and background context.
* Section 2 describes the general factors that affect the product and its requirements.
* Section 3 contains all the details the software developer needs to create the software.
* Section 4 defines technologies, infrastructure, performance, scalability, quality attributes, and other non-functional requirements.

# General Description

## Product Perspective

Bee Plan CSS is a desktop application that will be accessible from any computer with a Windows OS. The application is independent and self-contained, requiring no additional components beyond those mentioned above. It will be used by the user to manually integrate the common course schedules and the criteria provided by the Course Scheduling Coordination Office within the Student Affairs Office into the system. The program calculates a conflict-free course schedule for each department based on the integrated information and provides the output to the user.

## Product Functions

This product will have the functions below:

1. Receiving User Input: The system will allow users to upload data in the predefined (see Section 2.5 Assumptions and Dependencies, point 3) file format.
2. Data Integration: Once uploaded, the system will process and seamlessly integrate the provided data into its internal data structure, ensuring compatibility and proper functionality.
3. Generating Course Schedule: Based on the data provided by the user, the system will generate a conflict-free course schedule that complies with all predefined constraints and requirements, ensuring an optimized and efficient timetable.
4. Delivering User Output: The system will deliver the course schedule to the user in the specified file format, ensuring that the output meets all established standards and requirements for clarity and usability.

## Users and Characteristics

The Bee Plan CSS desktop application will be used by the course schedule coordinators of the relevant departments.

## General Constraints

### Technical Constraints

1. This product is a new and self-contained product that will be developed as a desktop application.
2. The program will be developed using the Java programming language.
3. The product's default language will be Turkish.

### Course Scheduling Constraints

1. Course loads cannot consist solely of graduate courses; they will be evaluated in conjunction with undergraduate courses (required or elective). It should be noted that projects, seminars, studios, and theses are not to be included in the mandatory course load.
2. In cases where classes, such as studio courses, have long durations, theoretical courses and studio courses should not be scheduled on the same day. For the Faculty of Architecture, graduate courses may be scheduled on days when studio courses are held.
3. Graduate courses should be scheduled in the evening hours (18:30 - 21:30).
4. No courses should be scheduled between 13:20 and 15:10 on Fridays, as this time is reserved for exams of common courses.
5. For each grade level, mandatory courses in the curriculum must not overlap. When scheduling class hours, it is up to the departments to decide whether 3-hour courses are scheduled as 2+1 or 3+0. Generally, mandatory courses are scheduled as 2+1, and elective courses as 3+0. Laboratory hours are scheduled for 2 consecutive hours and are placed after theoretical courses in the schedule.
6. The mandatory courses for third-year students will be scheduled in such a way that they do not overlap as much as possible with elective courses. When creating the elective course schedule, the schedules of the relevant departments will also be taken into consideration. (For example, the schedule of Computer Engineering elective courses will be considered for Software Engineering.)
7. Courses are scheduled in classrooms and labs that are available in the system. The number of participants is taken into account, with lab capacity not exceeding 40 students.
8. When creating schedules for all grades, the primary consideration should be the curriculum program for 1st, 2nd, 3rd, and 4th-year students, with separate schedules for each grade.

Faculty schedules for both undergraduate and other programs must be considered to avoid conflicts.

### Instructor Scheduling Constraints

1. Each faculty member should have a maximum of 4 hours of scheduled classes, including graduate courses, within a single day. Faculty members with a heavy teaching load may schedule their classes in a way that allows them to have one day free.
2. Faculty members with a heavy teaching load will first determine their undergraduate courses, and there will be no 4-hour limit for graduate courses.
3. In our university's departments/programs, requests for part-time assignments from outside the university should not be made unless the weekly teaching loads of full-time faculty members are distributed proportionally.
4. Faculty members' schedules must not overlap. A faculty member may be a member of the relevant department or may also teach as a guest from other departments or from outside the university. The schedules of instructors teaching departmental courses from outside the department should be obtained, and their classes must be scheduled in a way that avoids conflicts. For courses with lab hours and teaching assistants, the schedules of the assistants will also be included, ensuring that there are no conflicts.
5. For part-time faculty members, exceptions to day and time restrictions may be made when scheduling their courses, due to their coming from outside the university.
6. If a course is cancelled due to low student enrollment, the faculty member will make up for the resulting shortfall in their teaching load by taking on an additional course in the following semesters.

## Assumptions and Dependencies

1. Users are assumed to have basic knowledge of computer operations and software usage.
2. The application will depend on external data sets that need to be imported for proper functionality, such as the department curriculum course schedules from specific departments and common course schedule.
3. The format of the data files which will be imported from user and exported to user has not been determined yet. The application will support commonly used formats such as Excel (.xlsx) or Word (.docx), but the final decision regarding the acceptable formats will be made during the system implementation phase.

## Apportioning of Requirements

1. One of the secondary requirements, such as the user interface, will be developed in future phases of the project. Additionally, a wireframe representation of the user interface can be found in Section 3.1.1.

# The Specific Requirements

## User Interface

The user interface of the project will be developed to be simple, straightforward, and user-friendly, avoiding any unnecessary complexity or ornamentation. A clearly visible and self-explanatory button will be provided for users to upload data to the system. This button will be designed to prevent confusion or anxiety, ensuring an intuitive experience. Upon clicking the button, the system will open a file explorer for the user to select the required data. Additionally, an active process box will be implemented to maintain continuous communication with the user. This process box will provide real-time updates on the system’s status, including information messages, process updates, error messages, and potential solutions to any issues. Once the necessary processes are completed successfully, the system will open a file explorer prompting the user to specify the desired destination to save the formatted output data, which will then be saved accordingly. The following *(figure 1)* is a wireframe representation of the user interface.

metin, çizgi, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1

## Functional Requirements

In this section, the functional requirements for Project Bee Plan CSS will be outlined through system features and the key services offered by the product. The following use-case *(figure 2)* and activity diagrams *(figure 3)* illustrates the high-level interactions between user and the system.

metin, diyagram, çizgi, paralel içeren bir resim

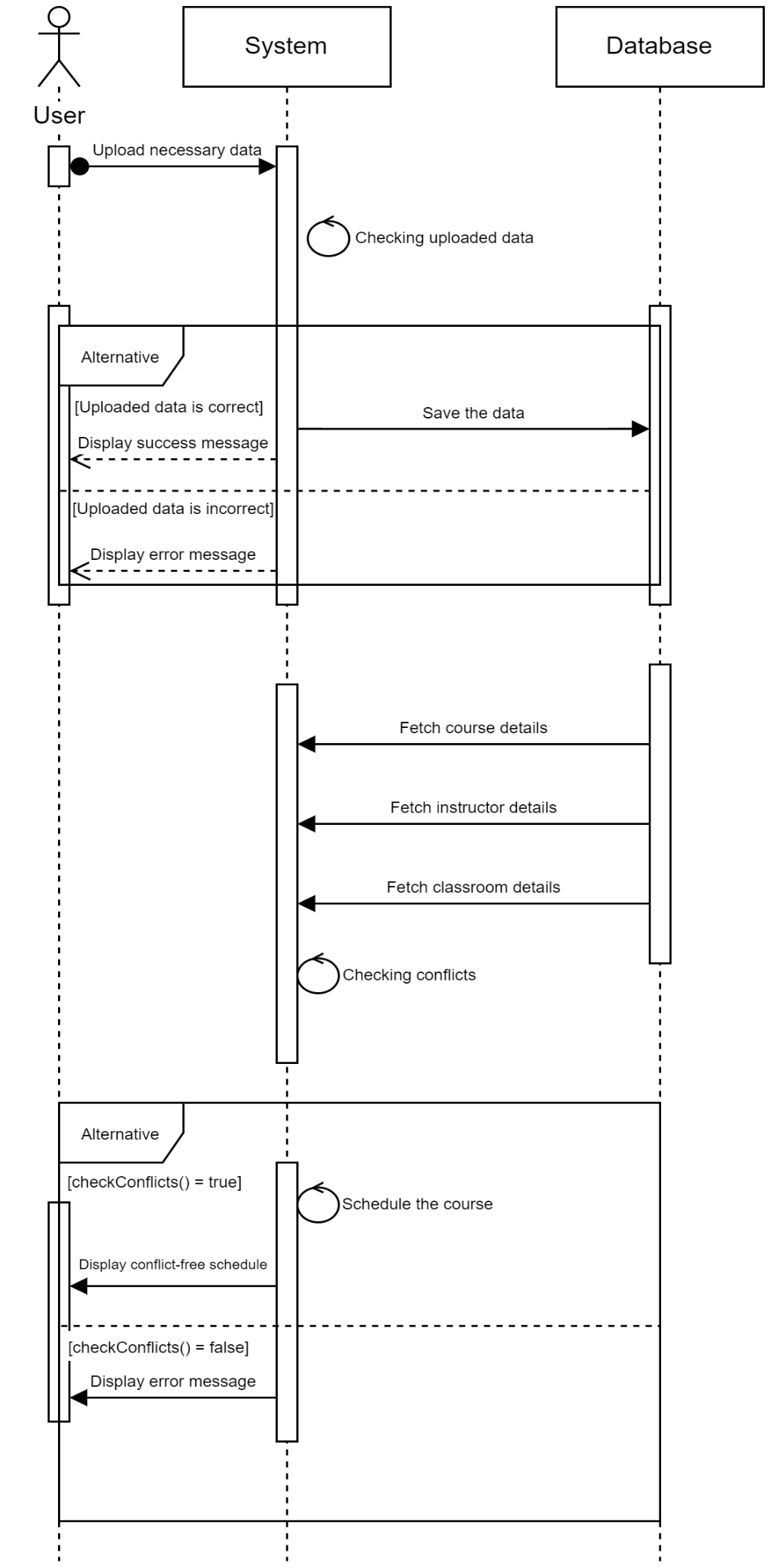
Açıklama otomatik olarak oluşturuldu

Figure 2

metin, diyagram, çizgi, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3



Figure

metin, diyagram, ekran görüntüsü, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure

### Receiving User Input

#### Description and Priority

The system allows users to upload data in a predefined file format, enabling the integration of course scheduling data into the application. This function is critical as it serves as the initial step for users to provide necessary data for schedule generation. This feature is high priority feature.

#### Stimulus/Response Sequences

1. User: Clicks the upload button.
2. System: Opens the file explorer for the user to select a file.
3. User: Selects a file in the predefined format and confirms.
4. System: Validates the selected file format.
5. If Valid: System proceeds to the next step.
6. If Invalid: System displays an error message: "Error! Uploaded file format is incorrect. Please check your file format."
7. User: Clicks to upload the file again (if an error occurred) or proceeds to data integration (if the file is valid).

#### Functional Requirements for Receiving User Input

REQ-1: The system shall provide a clearly labeled upload button for the user.

REQ-2: The system shall open a file explorer window when the upload button is clicked.

REQ-3: The system shall accept files in the predefined formats (e.g., Excel or Word).

REQ-4: The system shall validate the uploaded file format.

REQ-5: If the uploaded file format is invalid, the system shall display an error message indicating the issue.

REQ-6: The system shall confirm successful upload of a valid file to proceed with data integration.

### Data Integration

#### Description and Priority

This function processes the uploaded data and integrates it into the system’s internal structure, ensuring that all data is formatted correctly and compatible with the scheduling algorithms. This feature is high priority feature.

#### Stimulus/Response Sequences

1. System: Receives the uploaded file.
2. System: Processes and integrates the data into its internal structure.
3. If successful: System confirms successful data integration.
4. If unsuccessful: System displays an error message detailing the issue with the data.

#### Functional Requirements for Data Integration

REQ-1: The system shall read and parse the uploaded data file.

REQ-2: The system shall validate the structure and content of the data.

REQ-3: If integration is successful, the system shall notify the user of successful integration.

REQ-4: If integration fails, the system shall provide specific error messages detailing the problem.

### Generating Course Schedule

#### Description and Priority

Using the integrated data, the system generates a conflict-free course schedule that adheres to all predefined constraints and requirements. This function is vital for ensuring that the scheduling process is efficient and meets all academic regulations. This feature is high priority feature.

#### Stimulus/Response Sequences

1. System: Receives all integrated data.
2. System: Analyzes the data against predefined constraints.
3. System: Generates a conflict-free course schedule.
4. System: Notifies the user of the successful generation of the schedule.

#### Functional Requirements for Generating Course Schedule

REQ-1: The system shall apply all defined constraints while generating the schedule.

REQ-2: The system shall ensure the generated schedule is conflict-free based on the provided data.

REQ-3: The system shall notify the user when the schedule has been successfully generated.

### Delivering User Output

#### Description and Priority

The system delivers the generated course schedule to the user in a specified file format, ensuring clarity and usability in the output. This function is essential for providing users with a tangible result of the scheduling process. This feature is high priority feature.

#### Stimulus/Response Sequences

1. System: After generating the course schedule, prompts the user to select a destination for saving the schedule.
2. User: Selects a destination and confirms saving.
3. System: Saves the schedule in the specified file format.
4. System: Displays a confirmation message indicating successful saving of the schedule.

#### Functional Requirements for Delivering User Output

REQ-1: The system shall prompt the user to select a destination for saving the course schedule.

REQ-2: The system shall save the course schedule in the specified format.

REQ-3: The system shall confirm successful saving and inform the user of the saved location.

# Other Nonfunctional Requirements

## Performance Requirements

REQ-1: The system shall ensure that the response time for user actions (e.g., clicking buttons, uploading files) does not exceed 2 seconds under normal operating conditions.

REQ-2: The system shall maintain data integrity and accuracy during processing, ensuring that the generated course schedule reflects the input data accurately.

## Software Quality Attributes

REQ-1: The system shall demonstrate high reliability by maintaining continuous operation without crashes or unexpected failures. It should be capable of recovering from common errors gracefully without data loss.

REQ-2: The user interface shall be intuitive and user-friendly, allowing users to navigate and perform tasks without extensive training. It should provide clear instructions and feedback to enhance the user experience.

REQ-3: The system shall be designed with modular architecture to facilitate easy updates and modifications. Documentation shall be provided to support developers in maintaining and enhancing the software.

REQ-4: The application shall be compatible with any computer running a Windows operating system, ensuring easy deployment across different machines within the university.